ADAM

The Home Computing System

A Guide to an Extraordinary Machine
and Its Software

Eric N. Berg and Alan Smith

ADAM: The Home Computing System

It's All Here

Here is everything you need to know about ADAM, the sensational computing system from Coleco. ***ADAM: The Home Computing System*** will take you from the very beginning when you need to know how to assemble your new computer right through to sophisticated uses of the ADAM. Not only will this how-to book tell you how to make the most of ADAM's equipment and its accompanying software, but it's also a great reference that you'll turn to long after you're experienced with the machine.

ADAM: The Home Computing System covers:

- **ADAM's Equipment — How to put together your machine and set up your printer. How to handle data packs, ADAM's unique memory storage devices.**
- **ADAM's Operating System — How to create, store and retrieve files. How to make backup copies of your programs.**
- **SmartWriter — How to use ADAM's word processing program to edit, format and print out letters that look as though they were typed by an executive secretary.**
- **SmartBASIC — How to create your own programs with ADAM. The text also includes a number of practical home management and educational programs that will run as soon as you type them into ADAM.**
- **Graphics — How to create elaborate, full-color images on the screen using either low- or high-resolution graphics.**
- **Maintenance and Troubleshooting — How to keep your ADAM going and make minor repairs. The text even includes lists of common symptoms and their causes and how you can fix the problem.**

This comprehensive guide will start you out and keep you on the right track with all home computing systems. If you have an ADAM, then you

ADAM: The Home Computing System

Eric N. Berg and Alan Smith

0 00066 31936 01495

ISBN 0-88693-066-9

$14.95
A Banbury Book
Distributed to bookstores by
The Putnam Publishing Group
Cover printed in U.S.A.
ISBN 0-88693-066-9

FREE WISHBOOK!

MAIL THIS CARD TODAY!

Banbury Books is pleased to offer you our full-color catalog of computer books. Just fill out this postpaid card, send it to us, and we'll mail your catalog immediately.

ADAM: The Home Computing System

Name _____
(Please print)

Address _____ Apt. _____

City _____ State _____ Zip _____

Was your copy of *ADAM: The Home Computing System* damaged in shipment?

Yes_____ No_____

If yes, list damage _____

We'd appreciate your answering the questions below:

Primary use of your ADAM:

- ☐ Home
- ☐ Data base access
- ☐ Business
- ☐ Recreation
- ☐ School
- ☐ Word processing

Please check any other systems you own or use regularly:

- ☐ Commodore 64
- ☐ TRS-80
- ☐ Macintosh
- ☐ PCjr
- ☐ IBM PC
- ☐ IBM PC compatible _____
- ☐ Apple IIe
- ☐ IBM XT

Please list any computer users' groups you belong to: _____

_____

Age _____ Marital status _____ No. of children _____

Approximate family income:

- ☐ $10,000 - $20,000
- ☐ $20,000 - $30,000
- ☐ $30,000 - $40,000
- ☐ $40,000 - $50,000
- ☐ Over $50,000

Thank you.

No Postage
Necessary
If Mailed
in the
United States

BUSINESS REPLY MAIL
First Class Permit No. 114 Wayne, PA 19087

Postage will be paid for by Addressee

BANBURY BOOKS, INC.
353 West Lancaster Avenue
Wayne, Pennsylvania 19087

The Home Computing System

ADAM

The Home Computing System

Eric N. Berg and Alan Smith

A Banbury Book

Published by
Banbury Books, Inc.
353 West Lancaster Avenue
Wayne, Pennsylvania 19087

ISBN: 0-88693-066-9
First printing — July 1984
10 9 8 7 6 5 4 3 2 1
Printed in the United States of America

DISCLAIMER OF ALL WARRANTIES AND LIABILITIES

The authors and Publisher make no warranties, either expressed or implied, with respect to this book or with respect to the programs or the documentation contained herein, their quality, performance, merchantability, or fitness for any particular purpose. The authors and the Publisher shall not be liable for any incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or the use of materials in this book.

TABLE OF CONTENTS

INTRODUCTION

Buying a computer is often like buying a car. The list price describes a stripped-down model. Everything is optional. In the case of the car, optional features may include a radio, air conditioning, rust proofing or radial tires. With personal computers, the "extras" are much more necessary. You *need* a printer and the cable that hooks it to the computer. You need a way to store your programs and data: a tape or disk drive. And a computer can't run without software: you must have at least a programming language, and you probably want other programs as well, such as a word processor. As a result, when you walk out of the store with a new personal computer, you've probably spent considerably more than the list price.

But ADAM is different. For a single price (usually under $700), you get an entire computer *system.* ADAM comes with everything you need to get started. This remarkably easy to use machine comes with a built-in word processor, a version of the BASIC programming language, a printer that produces letters and other documents that look like they were typed on a high quality typewriter, even a game controller so that you can play video games on your computer. Just hook the ADAM up to your television set and your personal computer is ready to run.

Although the ADAM is simple for a computer, not all of its operations are obvious to beginners. If you're like most ADAM owners, this is your first personal computer. You need more guidance than you can get from the manuals that come with the machine. *ADAM: The Home Computing System* is a thorough introduction to the ADAM and all its features. As you leaf through this book, you'll find that nothing is left to chance. We start out with the obvious: how to put the system together. We tell you where to find the on/off switch. We walk you through ADAM's keyboard. Once you've finished the first two chapters, you'll understand ADAM's equipment, in and out.

One of ADAM's most practical features is its built-in word processor, SmartWriter. We devote two chapters to teaching you how to write and edit on a computer, including how to insert and delete words, and how to format a page. You'll learn how to save your correspondence in ADAM's data pack. Naturally, you'll also learn how to print out handsomely typed letters, resumes and reports.

ADAM: The Home Computing System can help you learn computer programming as well. The machine comes with a data pack of SmartBASIC, ADAM's version of the most popular programming language for beginners. This book will teach you programming logic, SmartBASIC commands and how to create graphic images on the screen. You also can try out the numerous BASIC programs listed in the book, including a loan amortization program, a program for computing how much a college education will cost, a check balancing program, a computerized address book, and much more.

Although the ADAM may be an incredible value in computers, it's still a machine. And like all machines, ADAM may break down occasionally. That's why we've included a chapter on maintenance and troubleshooting. Is the ribbon in your printer jammed? Does your ADAM sometimes refuse to save changes in your programs? We tell you what the likely causes are and how you can fix the problems. We also give you names, phone numbers and addresses of people you can contact if your ADAM requires major repairs. With a little preventive maintenance and some expert help, your ADAM should continue to run smoothly for years.

So, welcome to the new world of home computers! We hope you enjoy *ADAM: The Home Computing System* as much as you enjoy programming, playing and word processing with your ADAM.

CHAPTER 1
GETTING STARTED

Most computer systems are sold in bits and pieces. You pay separately for everything: the keyboard, the memory, the printer, even the cables that connect everything. But ADAM is different. ADAM comes to you as a complete computer system. You get not only the computer itself, but also a letter-quality printer, game controllers, programs and all the necessary cables and electrical equipment you will need to put the system together. Indeed, the great value of ADAM is that for one price you receive everything necessary to get started in amateur computing. And the best way to begin is to take stock of ADAM, making sure that all of the parts are there. Here's a checklist.

THE MEMORY CONSOLE The true brain of ADAM, the memory console comes packed with 80 kilobytes of working memory. (One kilobyte is 1,024 bytes.) About 54 kilobytes of memory are occupied by the operating system. About 26 kilobytes of memory, which would accommodate roughly thirteen pages of text, are available for your use. There is also a built-in high-speed digital data drive that ADAM uses to store your documents and programs and that you can use to let ADAM access a variety of applications and games. In addition, there is room for a second drive, should you decide to expand ADAM's memory.

THE SMARTWRITER LETTER-QUALITY PRINTER This may look like a typewriter—until you see it typing by itself at 120 words per minute! You can use regular, single sheets of paper or attached sheets of fanfold paper made especially for computers. If you're accustomed to using cartridge ribbons on a typewriter, you'll feel at home with ADAM's printer. The cartridge slips easily in and out, making changing ribbons a breeze.

THE KEYBOARD For under $1,000 you expect to get a toy keyboard with awkward clicking buttons instead of real keys, right? Not with ADAM. Your computing powerhouse comes with a professional, typewriter-style keyboard with 75 regular keys and 6 dedicated "function" keys—enough for any typing or computing task. You'll also find two joystick game controllers and a plastic game controller holder should you decide to use only one joystick at a time.

THE CORDS AND OTHER ELECTRONIC GEAR necessary for hooking up. In addition to the two cables that link ADAM's joysticks with the keyboard, your new computer system should come equipped with a coiled cord to link the memory console to the keyboard. You'll also find a three-pronged electric adapter and a small metallic box called a switch box that enables you to switch your television from its conventional use to use as a computer monitor.

THE SOFTWARE This is probably the best example of how ADAM scores a price breakthrough. While many other computer makers require you to pay extra for software programs, ADAM comes already equipped with some of the most popular computer programs available. You'll find enclosed a *digital data pack* with SmartBASIC on it, and another data pack with Coleco's newest science fiction action game—Buck Rogers: Planet of Zoom. And let's not forget SmartWriter—the powerful and versatile word processing system that is already built into ADAM and doesn't require a special data pack. Finally, you'll find a blank data pack to record programs, letters and other documents you wish to store on tape. Be sure to take good care of the digital data packs. They are very sensitive to electrical and magnetic fields. Never leave them on top of the console during use or in a data drive when switching the power on and off.

THE ADAM GUIDES To help you get to know ADAM, Coleco has provided you with three manuals. One, the set-up manual, will give you hints on setting up your machine and getting started. The other two guides discuss word processing and programming, respectively. Part of the purpose of this book is to put what Coleco has written into plain English that even the beginning computer user can understand.

In addition to making a highly technical subject easy to follow, however, in this book we also hope to teach you about other exceptional and heretofore unmentioned aspects of ADAM. For example, you'll learn how you can use ADAM to communicate with the outside world. We'll even explain how to draw color graphics with ADAM in exactly the same manner as advanced computer programmers, using computers costing thousands of dollars, do. To start, find a smooth, even surface on which to lay out ADAM's components. You're about to be introduced to one of the most exciting fields today—the burgeoning, fast-paced world of personal computers.

Now that you've made sure you have everything for your new computer system, it's time to hook up ADAM. Although this system has been designed so that all the parts fit together easily and quickly, it's important that you follow the assembly instructions carefully so that you don't cross any wires or leave important components disconnected. Here is a checklist of assembly steps.

1. Make sure that all electric wires are disconnected. This means disconnecting the plugs of your television set and of ADAM's letter-quality printer.

2. Plug the flat keyboard cord into the opening marked "keyboard" on the back of the memory console. Then take the other end of the keyboard cord and plug it into the slot at the back of the keyboard.

3. Find the two adjoining ports for ADAM's game controllers on the right side of the memory console. Plug the game controller cords into them.

4. Attach the game controller holder to the keyboard. Do this by hooking the holder under the keyboard and then sliding it away from you until it locks into place.

5. Snap the game controller which is plugged into the leftmost port into the holder.

6. Plug the SmartWriter printer cord into the side of the memory console. One end of the cord is already attached to the printer.

7. Hook up the antenna switch box to the television set that will serve as ADAM's screen. The antenna switch box—a small, rectangular, metallic device—enables you to switch easily between using your ADAM computer and regular television viewing. By using the switch box, you can avoid having to disconnect wires on your set every time you want to watch television instead of use ADAM. You just flick the switch on the switch box!

Unfortunately, the antenna wiring systems for television sets differ widely by manufacturer. At last count, United States television makers were using at least five different types of wires to attach their antennae. Such diversity makes it extremely difficult to describe all the possible ways that a television set might be hooked up. What follows, then, is a description of four of the most common ways:

7a. If a flat, double-wire, indoor antenna is used on your television set: With the television plug disconnected, use a screwdriver to remove the twin antenna leads from the back of your television. In their place, attach the twin leads from the switch box. Then take the television antenna leads that you have just disconnected and attach them to the other side of the switch box.

7b. If a flat, double-wire, outdoor antenna is used on your television set: With the television plug disconnected, use a screwdriver to remove the antenna leads from the back of your television. At the same place, which should be marked "VHF," attach the twin leads from the back of the switch box. Attach the outdoor antenna leads that you have just disconnected to the other end of the switch box.

7c. If you have a coaxial cable antenna on your television set: First remove the cable antenna from the back of your television. At this point, you may have to buy two special attachments: one that converts the round end on your antenna into two flat wire leads, and another that enables you to attach the switch box to your television set. Consult your local television repairman or electronics store salesman to see how this is done. Once you've gotten these special parts, connect the coaxial cable to the switch box at the place marked "antenna." Then connect the two leads from the switch box to the back of your television.

7d. If you have cable television: Essentially, follow the same procedure described in 7c. Once again, you'll have to buy some special parts to connect the switch box to the television and the cable to the switch box. Consult your local television repairman or electronics store clerk.

Now determine which channel you want to dedicate to ADAM. Your choice is limited to either channel 3 or channel 4. Choose the one that isn't used for broadcasts in your area, if such a case exists. Now, push the switch on the back of ADAM's console to the position corresponding to the channel you've selected. This switch is located next to where you plug in the cord from the antenna switch box.

8. Finally, check to see if the power outlet you plan to use has three openings or two. If the outlet is for a three-prong plug, attach the power cord from the back of the printer to the outlet, making sure beforehand that it is the standard 110 volts. If your outlet is for two-prong plugs, attach the adapter that comes with your ADAM system to the end of the printer power cord. Then plug the cord into the outlet. Congratulations! You've just completed putting ADAM together.

To Turn on ADAM

You've taken an inventory of ADAM's parts and you've worked diligently to assemble all of the components correctly. Now the moment you've been waiting for has finally arrived; it's time to turn ADAM on.

1. Set the antenna switch box to the setting marked "computer" by sliding the switch toward the top of the box in the direction of the word "ADAM."

2. Turn on the television, setting the volume at a comfortable level. Then turn the channel setting either to channel 3 or 4—whichever one corresponds to the switch that you previously set.

3. Find the on/off switch on the back of ADAM's printer. Push it to the "on" position. A title should appear briefly on your screen, to be replaced almost immediately by the screen used by ADAM when it is working as an electronic typewriter. At this point, ADAM has been turned on and is ready to work. To make sure of this, check that the red power light on the right of ADAM's keyboard is lighted. Indeed, whenever you're working with ADAM, this power light should remain bright.

If you have problems, check to make sure that all the connections and cables linking the components of the ADAM system are attached in their proper places. Next, make sure that your television set and ADAM's printer are securely plugged into their outlets. Finally, make sure that you've turned your television to the appropriate channel (3 or 4), and that the TV volume switch is turned to "on." Usually, going through this routine will enable you to get to the root of your problem. If you're still having trouble getting ADAM going, turn to the trouble-shooting section at the end of the book.

CHAPTER 2
PRESENTING ADAM

ADAM can be thought of as a series of components that together form a computer system. The parts work together to accept information from the outside world, process that information and then display the results in a way that you can understand. When separated, each part of ADAM is nearly powerless. But when combined, those same parts make up a powerful electronic tool that can perform a range of useful and exciting tasks, including word processing, home money management, entertainment and complex mathematical calculations.

The Memory Console

ADAM's center is the "computer" itself, known among more advanced users as the microprocessor. As the word implies, ADAM's microprocessor processes all information according to the instructions given to ADAM. More specifically, the microprocessor is the brain where all of ADAM's "thinking" occurs. For example, you can easily instruct ADAM's microprocessor to calculate the square root of five, move a block of text from one paragraph to another, or store your grandmother's favorite recipe for peach pie. With the appropriate commands, ADAM's microprocessor can perform all these tasks quickly and effortlessly. Regardless of our perception of a task's complexity, all tasks are equally simple to ADAM. Some merely require more time than others.

If you could see inside ADAM, its microprocessor would look like a small, black, plastic box about two inches long, half an inch wide, and a quarter inch high. A small piece of silicon encased in the black plastic has replaced the thousands of wires, switches and capacitors that made up traditional electric circuits. This remarkable sliver of silicon is called a "chip." As an added benefit of its small size, ADAM's microprocessor is much faster, cheaper, more energy efficient and more reliable than the bulky circuits of the past.

Another extremely important part of ADAM is its memory. It is here that ADAM remembers any instructions told to it or any information that it has been asked to save. ADAM's memory consists of two major categories. The first is memory you cannot alter; the second is memory you can.

The memory that you cannot alter is known as Read-Only Memory, or ROM. The contents of ADAM's ROM were placed there by the manufacturer. It generally contains information and instructions necessary for ADAM's operation. You cannot erase the contents of Read-Only Memory; even when you turn ADAM off, the contents of ROM are still there, waiting until you turn ADAM back on.

In addition, ADAM contains memory in which you can store information of your choosing. Computer scientists have dubbed this kind of memory Random Access Memory, or RAM. Some RAM is erased when you switch off ADAM; some is not. If you want ADAM to remember your instructions or any particular information, you must store the data you wish to save in the RAM that is not erased when ADAM is turned off.

Now that you know your RAMs from your ROMs, you are ready to learn about their applications in ADAM. Deep inside the memory console, next to ADAM's microprocessor, are many other smaller chips, each encased in a black plastic shell. These contain ADAM's internal memory, both ROM and RAM. ADAM's internal RAM is often called the working memory. This is where all of the instructions and information with which ADAM is currently working are kept. This is the only memory erased when you switch off the power. The amount of RAM is important because it limits the number of instructions ADAM can follow: the greater the amount of RAM, the greater ADAM's ability to work with large sets of instructions.

ADAM's internal ROM contains an important set of programs called the operating system. The operating system is so important, in fact, that ADAM could not function without it. Of course, you cannot lose the contents of ADAM's ROM unless you take ADAM apart.

The cartridges you plug into ADAM's digital data drive and game cartridge slot also contain ROM memory, called external ROM. You can access the information and games in these cartridges as soon as you insert them into ADAM.

To save information that you have typed into ADAM, or the results of ADAM's computations, you must tell ADAM to send the information to the digital data drive, which contains a form of RAM that is not erased when ADAM is shut off. The digital data pack resembles an audio cassette tape, and is inserted into the drive to record the information.

Computer scientists measure the amount of a computer's memory in units called bytes, which are the largest units of information that ADAM's microprocessor can digest at any one time. ADAM often seems to read

great quantities of data all at once, but it is actually reading small pieces of data extremely quickly. Since a byte is not very large, computer users add the prefix "kilo" to the word "byte" to get the word "kilobyte." A kilobyte, abbreviated as "K," is 1,024 bytes. ADAM has 80K of internal RAM, enough to hold a document about forty pages long.

The Keyboard

The first thing you'll notice about ADAM's keyboard is that it's very similar to the keyboard you'd find on a typewriter. All the letters are in their normal places. The carriage return key is just where you'd expect it, on the right side of ADAM's keyboard, and the shift keys (which capitalize letters) appear on either side of the keyboard.

However, there are some other keys that are probably unfamiliar to you. These keys enable you to use ADAM both as an electronic typewriter and as a powerful word processor using the SmartWriter program. Some of these keys enable you to print text using ADAM's letter-quality printer, as described below:

1. Smart Keys

These are the six large, black keys along the top of ADAM's keyboard that are labeled with Roman numerals. They correspond to six colored rectangles that will appear on your television screen when you use ADAM. Since the rectangles are also numbered, you can easily tell which one corresponds to which Smart Key.

When you turn ADAM on, two colored rectangles appear on the screen. One, Smart Key V, is labeled MARGIN/TAB/ETC. The other, Smart Key VI, is labeled MARGIN RELEASE. Press Smart Key V. Suddenly, six rectangles appear, along with four new labels, HORIZ MARGIN, TAB, LINE SPACING, and DONE. Now, press Smart Key II, "HORIZ MARGIN." The screen changes again, and the labels LEFT, RIGHT and DONE appear. At this point, it's possible to change the left and right margins of your electronic typewriter with the arrow keys on the lower right part of the keyboard. By pressing Smart Key VI, you can return to the original screen, consisting of the rectangles labeled MARGIN/TAB/ ETC and MARGIN RELEASE.

This is only one example of how the Smart Keys are used. When you're using ADAM as a word processor, the Smart Keys have different functions. They enable you, for example, to delete text, move blocks of text, or change the colors on the screen. No matter how you're using

ADAM, every time you press a Smart Key the labels on your screen change. These new labels are messages to you. They guide you step by step, telling you what to do next or what will happen when you press a Smart Key.

2. Command Keys

These are ten keys of standard typewriter size, two in the upper left corner of the keyboard, eight in the upper right. Just as the Smart Keys perform diverse functions, so do the command keys. Moreover, the labels on your screen change when you press a command key, just as they do when you press a Smart Key.

Look first to the left, and you'll see the key marked ESCAPE/WP. By pressing this key after turning ADAM on, you convert ADAM from an electronic typewriter into a word processor, thus the abbreviation WP on the key. If you press ESCAPE/WP at any other time, you will be able to cancel a command you've given that ADAM has not yet executed. If you haven't given a command, pressing ESCAPE/WP another time won't do anything.

Now consider the key marked WILD CARD next to ESCAPE/WP. For now, this key does absolutely nothing. It has been reserved by Coleco for future use in as yet undeveloped programs. Therefore, we won't spend any more time explaining what this key doesn't do.

Now look to the right of the Smart Keys. The eight command keys there function only when using the SmartWriter word processing program, which you entered when you pressed ESCAPE/WP. Nonetheless, these keys are extremely important, as they permit you to undertake such diverse word processing tasks as moving, copying or deleting text, inserting letters or words into text, or storing text for later review. When you've made all necessary corrections on your text and are satisfied with the result, you can print it at 120 words per minute on ADAM's letter-quality printer with the PRINT key.

Notice the arrows set in the shape of a compass on the bottom of the keyboard. These are known as cursor-control keys, named after the small, white line that moves across your screen as you work. In computer language, this speck of light, known as the cursor, tells you where you are in a document. To move the cursor, simply press any one of the arrows, and the cursor moves in the direction indicated.

The HOME key, located in the center of the arrows, makes controlling the cursor particularly easy. When you press HOME by itself, the top line of text on your screen moves onto the roller (the black rectangle near the bottom of the screen), and the cursor immediately moves to the beginning of that line. If you press this key a second time, you will move the cursor to the beginning of the document. If you press HOME in

conjunction with an arrow, the cursor moves swiftly in the direction of the arrow. For example, if you press HOME and the right arrow, the cursor moves immediately to the right edge of your text. Try it. Pressing HOME and the up arrow moves the cursor to the top of the roller and moves the top line of text on the screen onto the roller. Finally, by pressing the down arrow and HOME together, it's possible to move the text below the roller onto the screen.

One final note: It's also possible to position ADAM's cursor by using the joystick on the game controller. If you push the controller's circular knob right, the cursor moves to the right. Push the knob down, and the cursor moves down. It's very much like steering a toy car. ADAM's cursor will move obediently in whichever direction you push the knob.

Exploring ADAM's Screen

If you've ever used a typewriter, chances are word processing will be fun and easy right from the start with SmartWriter. That's because Coleco, the company that invented ADAM, has designed ADAM's screen to resemble a typewriter. As soon as you connect ADAM to your television set and turn the computer on, most of what you see on the screen should be familiar.

The first thing that will strike your eye when you turn ADAM on is the long, black rectangle at the bottom of the screen. This is called ADAM's roller, and it functions exactly like the roller on a conventional typewriter—it is the area in which ADAM's "keys" strike. The rest of the screen can be thought of as the remainder of your paper. Although you can see the rest of the page, only by scrolling it onto the roller can you work with it. This is similar to moving paper back onto the roller when making a correction with a typewriter. In this sense, then, ADAM's roller can be thought of as your work area.

There is, however, one important difference between the roller on an everyday typewriter and ADAM's roller. Whereas a traditional roller is wide enough to hold a full page—that is, you can type the full width of a paper with a regular roller—ADAM's roller is only 36 characters wide. What this means is that it's not possible to fit a full line of printed text on one line of your television screen. Only 36 characters will fit. If you want to type further, you have to go to the next line on your screen. But the important point to remember is that this second line on your screen still corresponds to one printed line on a page. In fact, Coleco has designed

ADAM so that all the text that you can fit onto ADAM's roller corresponds to one line on a printed page. Not until you've filled up a roller, and your text scrolls up, have you in fact typed a full line.

Another way to think about this is as follows. Imagine you've just typed the following line of text on a regular typewriter: ADAM is an inexpensive, easy-to-use home computer from Coleco Industries. Now imagine cutting the text into three segments as follows:

> ADAM is an inexpensive,
> easy-to-use home computer
> from Coleco Industries.

By placing the middle segment below the first one and the last segment on the bottom, you have in effect followed the same method that ADAM does for typing text. Each of your narrow lines corresponds to one line on ADAM's screen. Your three lines together compose one printed line. In the same way, a full roller of type on ADAM corresponds to one printed line on a page. Not until you are done filling ADAM's roller with text have you in fact filled a regular typewriter text line.

Try the following. After turning ADAM on, press ESCAPE/WP to activate the SmartWriter word processor. Then type the following:

> When in the course of human events, it becomes necessary
> for one people to dissolve the political bands which have
> connected them with another, . . .

You'll notice that in this introduction to the Declaration of Independence everything up to the word "for" can be fit onto ADAM's roller. In other words, everything up to "for" corresponds to one printed line. As you type further, your text scrolls up onto the main part of ADAM's screen, and a clean line appears on the roller, enabling you to begin a new line of type—and a new printed line.

A question immediately arises. How can you tell where you are on a printed page? After all, the second line on ADAM's roller generally corresponds to the middle of a line on a piece of paper. How does one avoid being confused?

The answer lies with the speck of light that moves across the top of your screen as you type. This narrow white beam is called the "horizontal gauge." The horizontal gauge is an indicator of where you would be on a printed page. Since the scale across the top of the screen is a full 80 characters wide, you can tell exactly where you are by watching the position of the gauge. As you type, the gauge moves horizontally to the right, indicating that the text you are typing will be printed at that position when you eventually tell ADAM to print it. If you move the cursor left, the

gauge moves left. When the gauge approaches the right margin (indicated by a red bar along the right side of the gauge scale), a warning bell signals—exactly as it would on a regular typewriter. Therefore, you'll probably find it most helpful to concentrate on the horizontal gauge, rather than on the text in the roller, if you're trying to determine where you are on a printed page.

You'll also find it helpful to keep track of the white speck of light moving along the left side of your screen. This is the vertical margin marker and it tells you how far down the printed page you are. Every time you strike ADAM's RETURN key, the white speck moves down a line, indicating that you have begun a new printed line. When you scroll to the top of your document using the cursor-control keys, the white speck moves up. As such, it's easy to see when you're approaching the top of the page—and to avoid typing too close to the bottom of the page. Finally, the number at the top of the vertical margin marker indicates what type of paper ADAM thinks you're using: letter (11 inches long) or legal (14 inches long) size. It's also possible to change all four margins, the two horizontal ones and the two vertical ones. That will be covered in greater detail later.

The Printer

ADAM's printer, called the SmartWriter Letter-Quality Printer, is essentially a typewriter without a keyboard. Rather than typing directly on SmartWriter, you type your text into ADAM. ADAM then sends your text to SmartWriter when you tell it to print. This gives you the opportunity to edit your text before ADAM prints it. SmartWriter can print at about 120 words per minute, much faster than most professional typists. But unlike typists, ADAM rarely makes mistakes. The text will be printed on SmartWriter exactly as you typed it into ADAM.

As with most modern typewriters, you can change the style of the type, or font, of SmartWriter by replacing its *daisy wheel*—the thin, round piece of plastic about four inches in diameter with petals like a daisy's—with one that has the typeface you want.

How You Can Use ADAM

The first electronic computers that appeared in America in the mid 1940s were big, clumsy machines that had limited applications. Slow and costly, the first computers were used by the federal government, by large corporations, and by a few wealthy universities. The government, for example, used a computer called Univac I in census taking in the 1950s. Corporations soon began to use the machines to keep track of accounting information, and eventually, high schools and colleges used computers to perform high level mathematical calculations and to store student records.

Today, however, all that has changed. Computers, once reserved for large organizations, have now become a tool of ordinary people and are being used in some ordinary, yet very practical, ways. For example, many housewives are storing recipes on home computers; both teenagers and executives are keeping computerized electronic date books that can be updated quickly and neatly; and thousands of Americans have started writing their diaries on computers with word processors.

ADAM is one of the new computers now available for home use. It has been given certain basic abilities, enabling you to perform immediately such tasks as programming, word processing and game playing—the tasks all modern computers are expected to do. Additionally, by buying other programs available for ADAM on digital data packs, you will be able to perform such applications as money management, home budgeting, filing and creating an electronic diary and date book.

Programming

BASIC (Beginner's All-purpose Symbolic Instruction Code) is the programming language that virtually all novice computer users first learn. ADAM comes equipped with its own version of BASIC, called SmartBASIC. To begin writing your own programs, you need only to insert the SmartBASIC cartridge into ADAM's memory console. You must write your instructions to ADAM in SmartBASIC, because ADAM cannot understand everyday English. Fortunately, using SmartBASIC is very easy. Most SmartBASIC instructions will be familiar to you, since they are made up of simple English words.

Although SmartBASIC is simple to use, it is an advanced computer language. With SmartBASIC, you could program ADAM to compute the sum of every even number from 1 to 5,000, and ADAM could do it in only

a few seconds. Or you could instruct ADAM to alphabetize a huge mailing list in a few minutes—a job that could take you hours to finish. You will quickly discover that ADAM can save you from a great deal of hard, tedious work.

Finally, SmartBASIC can be used to teach children skills such as spelling and reading. With a brief program, you can teach a child to distinguish "cat" from "cap" and "coat" from "moat." Such computer-based learning is already under way in elementary schools nationwide. With ADAM and SmartBASIC, you can bring the newest learning techniques into your home.

Word Processing

Word processing has been described as both a writer's dream and the answer to an editor's prayer. What else enables a writer or an editor to delete or insert words into a text without leaving any mess, or to move entire blocks of text from one part of an essay to another without using scissors and paste? In a nutshell, that is the purpose of word processing, and ADAM's built-in program makes this as easy as using a regular typewriter.

To begin with, the cursor control keys on the right of ADAM's keyboard enable you to move anywhere in your text much more easily than moving to the appropriate place on a sheet of paper in a typewriter. Moreover, many of the keys at the top of ADAM's keyboard have been preprogrammed to insert, delete, and move text, and as more and more programs are written for ADAM, it may even be possible for you to create your own unique word processing commands. For example, you may be able to program ADAM's IV function key to erase whole paragraphs with a single keystroke. Then, every time you press IV, ADAM will delete the paragraph you no longer want.

With ADAM's word processor, you will be able to revise what you have written electronically and instantly display the revised version on your television screen. You will be able to store it in ADAM's memory and come back to it later to weed out errors. Finally, after you have made all your corrections, you will be able to select a format for your document using some of ADAM's other commands, and print the whole thing on ADAM's letter-quality printer. After using SmartWriter, you will never want to use a mere typewriter again.

Games and Entertainment

So far, you've learned about some of the more serious applications for ADAM. However, ADAM does more than hard work; many people purchase the machine to play electronic video games like those found in arcades—action-packed games like Pac-Man, Space Invaders, Donkey Kong and Zaxxon. Additionally, ADAM can play far more complex games. Adventure, for example, leads you through an underground maze filled with danger, intrigue and wealth, and chess challenges your mind rather than your skill with a game controller.

ADAM comes with its own arcade-style game called Buck Rogers: Planet of Zoom. The object of this fast-paced, three-dimensional space game, which comes in a cartridge that you plug into ADAM's game cartridge slot, is to destroy the engines of an enemy ship. You then enter the ship and emerge later with a more advanced ship of your own. At any one time, there can be as many as thirty-two moving objects on the screen—missiles, space ships, satellites and meteors. An alert mind and a quick hand are necessary to win this game.

A broad range of other game cartridges, from beginner's to advanced, is already available. In fact, any cartridge that can be played on Coleco's home video system, COLECOVISION, can be plugged into ADAM. Any of these games can take advantage of ADAM's 16-color graphics, keyboard and game controllers.

Money Management

Until recently, Wall Street investment experts and finance professors were the sole users of financial models. They programmed their big computers to model, or make various pictures of, investment opportunities. The outcome of such a picture depended on certain assumptions the modeler made, the level of sales or the tax rate, for example.

However, financial modeling is now available for nearly everyone. With ADAM it is no more complex than inserting a data pack into the memory console and telling ADAM what to do. Suppose you are buying a home and want to know how your monthly payments might vary with different mortgage rates. ADAM's SmartFiler spreadsheet program lets you change the interest rate and get the answer almost instantly. Suppose you want to compare monthly auto payments with various down payments. ADAM's spreadsheet program will quickly calculate the answer to this problem too.

Finally, we cannot forget one of the most popular uses for computers like ADAM, investing in the stock market. No one has a sure method for picking winning stocks, but thousands of armchair investors are being aided by home computers. ADAM can help you with your investments. ADAM's graphic abilities and calculating talents might make the difference between a good day and bad day on Wall Street.

CHAPTER 3
WORD PROCESSING

Crumpled first drafts. Typos. Correction fluid. Erasers. Chances are these terms are familiar if you've ever used a typewriter. How many times, for example, have you typed the first version of a letter only to find yourself tearing the paper out of the typewriter because you were dissatisfied with the results? Similarly, unless you're 100 percent accurate when typing, chances are you've used correction fluid to fix typos. Tearing up rough drafts and painting over errors can be time consuming, not to mention frustrating, and anyone who composes text on the typewriter, making corrections and revisions along the way, knows how cumbersome that can be. Fortunately, word processing provides a solution.

Basically, using word processing is the same as using a conventional typewriter—with one crucial difference. Whereas a typewriter transfers what you write directly to paper, a word processor like ADAM's SmartWriter displays your text on the screen. Rather than putting ink on paper to create letters and words, ADAM's SmartWriter creates letters and words electronically on your television set. The wonder of word processing is that once you have your text on the screen, it's easy to make corrections and revisions.

An example should illustrate how easy making corrections and other changes can be with ADAM's SmartWriter. Suppose you've written the word "accommmodate" with three "m's" instead of the two "m's" most people are used to. If you've used an everyday typewriter, you might be able to paint over the superfluous letter with correction fluid. For a cleaner look, you might use erasable typing paper and erase the rogue "m." With SmartWriter, by comparison, you need only position ADAM's cursor to the right of the extra letter and press ADAM's backspace key; the letter disappears before your eyes. That is why computer users refer to this as a "destructive backspace." In fact, if you hold down ADAM's backspace key, characters will continue to disappear until you release the button. With a single keystroke, you can delete letters, words, lines or entire blocks of text.

It's equally easy to insert text with SmartWriter. For example, suppose you've just typed the introduction to Lincoln's Gettysburg Address as follows: "Four score years ago, our fathers brought forth on this continent

a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal." Suddenly, you realize you've blundered—you've omitted seven years. The accepted version of Abraham Lincoln's speech begins, "Four score and seven years ago . . ." If you made this mistake on a typewriter, chances are it would be an irreparable error, and you would have to retype the page, but not so with SmartWriter. Using a series of commands that make up ADAM's insert function, it's possible to insert the missing words without retyping anything. What's more, the rest of your document automatically adjusts to the insertion, creating just enough blank space in which to set the inserted words properly. In fact, with ADAM's insert key it's possible to insert whole words and paragraphs anywhere in your text, exactly as if you had written them there in the first place.

Finally, consider the times when you've wanted to move or reorganize text while writing. The traditional way to do this is by using scissors to cut the text into pieces and paste the rearranged sentences on a new page. Again, just as you'd expect, SmartWriter provides a cleaner, quicker alternative to "cutting and pasting." This alternative is ADAM's MOVE command, which permits you to move a highlighted block of text anywhere in the document without running through the house looking for the good sewing shears.

These are just a few of the time saving ways in which you can use SmartWriter. We'll be telling you more about these time savers in the chapter on advanced word processing. For now, however, we'll leave you with a parting thought: Most people who use word processors find that they never want to use typewriters again. They find the convenience of editing text "on line"—that is, on a computer screen—more efficient (and fun!) than working with any electronic typewriter. Chances are that after getting to know SmartWriter, you'll want to put your typewriter into storage too.

Getting Started With SmartWriter—Entering Text

Getting started with SmartWriter couldn't be easier. To begin, simply turn ADAM on by flipping the power switch on the back of the printer. At this point ADAM functions as an electronic typewriter, with everything you type on ADAM's keyboard immediately transmitted to the printer. To get started with SmartWriter, press the ESCAPE/WP key.

You'll notice that the Smart Key labels at the bottom of the screen change. These labels guide you through the process of changing margins and tab stops, altering the color of the screen, erasing and inserting text, and searching for characters and words in the text. However, you can begin working with SmartWriter without using any of these functions, since Coleco has preset margins and tab stops for you. Let's get right into things by entering some text.

Type the following proverbs:

A bird in hand is worth two in the bush.
A stitch in time saves nine.

Don't worry about typing too fast. ADAM's electronic brain can work so quickly that you should not be able to overload it. You certainly don't need to worry about correcting mistakes; you'll be able to fix them easily later. For now, simply type the sentences and watch what happens when you reach the end of the line on the roller.

If all goes well, you won't have to press the return key. Instead, ADAM senses when you're approaching the end of a line and automatically skips to the next line at the right moment. Moreover, if ADAM can't fit a word at the end of the first line, it automatically shifts that word to the next line. This feature, called *word wraparound,* makes ADAM's return key used far less frequently than in regular typewriter use. With ADAM, about the only time you truly need to hit <RETURN> is to end a line before you reach the right margin or to begin a new paragraph. Otherwise, you can type to your heart's content, leaving the creation of new lines to ADAM.

Storing Text for Future Use

Suppose you've just finished typing your first novel, but you're not ready to print it out. Instead, you'd like to leave ADAM for a while, muse over what you've written, then make revisions after you've returned from a show or the pizza parlor, or some other diversion. At this point, it's necessary to record your cherished prose on the blank digital data pack that comes with ADAM. The data pack acts as ADAM's memory, capable of recording everything you've written. It looks like an ordinary audio cassette, but you won't be able to insert it into your cassette player because the placement of the holes in the plastic case is different. Similarly, a normal cassette can't be loaded into ADAM.

To store text, insert the data pack into the tape drive in ADAM's memory console. Be sure *not to* put in the data pack marked SmartBASIC or the one containing a game. Next, press the STORE/GET command key on the right side of ADAM's keyboard. Suddenly, new Smart Key labels appear at the bottom of your screen. One, labeled STORE SCREEN and numbered IV, will enable you to store only what's on your screen. If you want to store your whole document, press Smart Key V, labeled STORE WK-SPACE. The expression WK-SPACE stands for "work space," and it refers to the entire document with which you are working. For now, we'll record only what's on the screen by pressing Smart Key IV.

After you do this, the labels at the bottom of your screen change again. This time, ADAM wants to know on which tape drive you'd like to store your text. If you have two tape drives, you can choose between drives A and B. Most persons just starting have only one drive—in the A slot. We'll assume that's your position too. Press Smart Key III, labeled DRIVE A.

The labels on the screen change yet another time, and ADAM asks you to give your novel a name so that it can easily retrieve your masterpiece from memory when asked. This name, the filename, will be used to refer to this document when you retrieve it. It can be up to ten characters long. You may use only letters, numbers or spaces in the filename. In the lower left region of the screen, you will see the following printed:

```
NEW FILE
PLEASE TYPE
NAME OF FILE
```

Whenever you see the NEW FILE message, you must use a filename that has not been used before. You will not be able to store the same document twice using the same filename. This may be changed in subsequent versions of SmartWriter, since it prevents you from storing your document as you are writing it. Fortunately there is a trick you can use, which we'll discuss in the section on retrieving data. When you have typed in a valid filename, press Smart Key VI, labeled STORE SCREEN. At this point your text should disappear, the tape drive should begin whirring, and ADAM should flash a new message on the screen informing you that it is busy storing your text. Don't worry if the tape seems loud or if it is spinning quite rapidly; this is part of ADAM's advanced memory system. Rest assured that your prized prose is being etched onto the tape. You'll know that ADAM has finished recording your work when your text reappears on the screen and the original six Smart Key labels return as well. It's now possible to pick up where you left off or take time out to ponder your prose. To do so, remove the data pack and turn off the switch on the back of the printer.

Retrieving Stored Text

You're back. You've spent hours thinking about ways to improve your novel. Now you've returned to ADAM and are ready to retrieve your "manuscript" to make the necessary revisions. How is it done? Turn on ADAM and place the data pack on which your novel is stored into drive A. Press the ESCAPE/WP key and then the STORE/GET key.

As before, the Smart Key labels change, only this time instead of pressing Smart Key IV to store a screenful of text, you press Smart Key VI, labeled GET. The labels change once more, and ADAM asks you from which tape drive you'd like to get material. Press Smart Key III, labeled DRIVE A.

At this point, ADAM tells you to be patient while it retrieves the directory for drive A. The directory is exactly what you might expect—a listing of the names of documents that have been stored on the tape in A. Each filename appears on what look like the tabs of file folders. To retrieve your document, position the arrow on the file folder beside the correct filename. You can move the arrow by pressing the appropriate cursor-control keys on the bottom of the keyboard. With the arrow pointed at the appropriate filename, you're ready to retrieve your novel from memory. Press Smart Key VI, labeled GET FILE, and the tape will whir in search of your pride and joy. Moments later, the first few lines of your novel should reappear on the screen, along with the original six Smart Key labels. The remainder of your document is in ADAM's working memory, its work space. You can move the downward arrow of the cursor-control keys to make the entire document scroll up the screen.

Always remember that when you retrieve files, ADAM doesn't erase the file on which you are currently working. If you've been typing a new document, or have previously retrieved one, both documents—that which you see on the screen (which is in ADAM's working memory) and the one you just told ADAM to get—will now be in ADAM's working memory, one following the other. You will have merged both documents into one new one, now stored in ADAM's working memory. This is a useful feature if you want to merge files, but a headache if you don't.

If you don't want to merge the documents, erase the document that is in the working memory (or store it on a data pack, if you don't want to lose it) before telling ADAM to get another document. Use the procedure in a following section, Clearing the Screen and Work Space, to erase the document. Of course, if you've just turned ADAM on and haven't begun typing, you won't have anything in ADAM's working memory to erase.

Once you've retrieved your novel, it's possible to make corrections to what you've written or to add to the text. However, when you're done with the revisions and are convinced you have a winning yarn, you still have another important decision to make. You can either record the new

version over the old, in effect erasing the old version, or else create a separate file for the new version, leaving the old one intact. To make this choice, you have to go through the procedure for storing files (outlined above). This time, however, the message is slightly different when you store the document. Instead of NEW FILE, you will see FILE: followed by the name of the file you just retrieved. If you don't type in a new filename and press Smart Key VI at that point, you will store the file on which you are currently working in place of the old file with that filename. If you don't want to do that, type in a new filename.

Here is the promised trick. After retrieving text from the digital data pack, you can store your document any number of times as long as you don't type in a new filename. If you are planning to create a long document, consider storing it after you have completed only the first few lines. Then clear the work space (using the instructions given previously) and retrieve your document. Now you can continue. Store the document each time you finish several pages. This will prevent you from losing too much text in case of an unforeseen event, such as accidentally pulling out ADAM's plug.

When you store a new document with the filename of an old one, the old one isn't erased or lost immediately; it's transferred to another directory, a *backup* directory. You can access this directory when you see the file directory on your screen during the process of retrieving data. Just press Smart Key V, labeled BACKUP FILE DIR. and the backup directory will appear in the regular directory's place. At this point you can retrieve any of these files.

If you store a document with the same filename three times, however, the oldest version of the file will be lost. The backup directory saves only the version of the document that was in the regular directory just prior to using a STORE command.

You can also delete filenames from the directories with SmartWriter. You might want to delete old files so you won't run short of storage space if you have a limited number of digital data packs. To delete a file, tell ADAM to show you the directory of files that are stored on the digital data pack. You can do this with the STORE/GET key, following the procedure necessary to save files or to bring a document into ADAM's working memory. When you see the directory, just move the arrow to the name of the file you want to erase and press the DELETE command key. ADAM will then make you confirm this request by pressing Smart Key VI. If you change your mind at this point, just press the ESCAPE/WP key. Similarly, you can delete files that are stored in the backup file directory by telling ADAM to display that directory on your screen.

Learning to Correct Your Mistakes

SmartWriter can make basic changes to your text, such as deleting and inserting material, and can also perform advanced tasks, such as moving and copying blocks of text or searching for a particular word or expression in a body of text.

To begin, enter SmartWriter and type the following text:

A penny saved is a penny earnez.

We've intentionally misspelled the word "earned," placing a "z" on the end instead of a "d." To correct this error on a conventional typewriter, you'd have to break out the correction fluid and paint over the mischievous "z" or erase it by rubbing vigorously and maybe tearing the paper—but not with SmartWriter. With the cursor positioned just after the "z," try pressing the backspace key and watch what happens. The "z" disappears. It is gobbled up by the cursor. If you press BACKSPACE again, the "e" will be erased from your screen. Indeed, as long as you press BACKSPACE, the cursor will continue moving left, swallowing everything in its path. It's then possible to correct what you've written simply by typing forward.

In fact, SmartWriter has been designed so that you don't even need to erase letters to correct them—you can type right over them. In the example above, it's possible to move to the beginning of the word "earned" without erasing any of the letters. You simply move the cursor left using the cursor-control keys, and "earnez" remains intact. Now, with the cursor at "e," type "earned." Sure enough, you've written over every letter, as if the first word didn't exist. And the miscreant "z" should be vanquished; you've changed "earnez" to "earned."

Using the Delete Key

If you're like most typists, there will be times when erasing a single letter or word won't be sufficient—you'll want to perform radical surgery on your writing by deleting entire blocks of text. Suppose, for example, that you've typed the following sentence:

In 1492, Columbus sailed the ocean blue blue.

Was the ocean so serenely blue that it's necessary to type "blue" twice? Probably not. So let's delete the last word using SmartWriter's delete procedure. Here's how:

Start by pressing the command key on the right side of the keyboard marked DELETE. You'll notice that the Smart Key labels on the bottom of the screen change. At this point you need to indicate to ADAM the block of text you'd like to erase. This process of identifying a group of words for editing is called *highlighting,* and with ADAM it's achieved by underlining the words in question.

In our sample, we want to erase the word "blue," so bring ADAM's cursor under the letter "b." Now press Smart Key IV, labeled HI-LITE. The label should immediately change to HI-LITE OFF, meaning that until you press that Smart Key again, ADAM's highlight, or underline, function is on.

Begin moving the cursor right. First move it under the "l" in "blue," then under the "u" and then the "e." If all goes well, ADAM should underline the entire word "blue." You'll know this is happening because a red line will appear under the word to be deleted. And, as is always the case when deleting text, you'll want to highlight the blank space after the word you want to eliminate also. Therefore, extend the highlighting beyond the letter "e" by one space. This way you won't end up with extra space between words when you're done.

Once you are finished highlighting, tell ADAM you're done by pressing Smart Key IV, labeled HI-LITE OFF. You've now marked the word to be deleted and turned off ADAM's highlighter. There's only one more step: actually deleting the word from your screen. Press Smart Key VI, labeled FINAL DELETE. Sure enough, "blue," the underlined word, disappears from view. The text to the right of the deleted word will move over toward the left. Nothing will remain to show that the extraneous word had ever existed, not even a gap. It would be as if you had never made the typing error.

Inserting Text

Sometimes, of course, you want to insert rather than delete text. If you want to insert text using a typewriter, you have the choice of either starting over or scrolling up a line and placing your insertion over the sentence in which you're working. Neither provides a particularly elegant solution. Fortunately, SmartWriter does.

Suppose you're a rock music fan, and you've just typed the following sentence from a Beatles song:

It's been a hard day's night, and I've been working like a dog.

Being an award-winning songwriter, you decide to improve upon the Beatles' work by inserting the words "so long" after "working." In other words, you'd like the sentence to read: "It's been a hard day's night, and I've been working so long like a dog." With SmartWriter, this can be done easily, as follows.

First, place the cursor exactly where you'd like to make the insertion. In this case, you'd like to make the insertion at the word "like," so place ADAM's cursor under the "l." Second, press the command key at the top of the screen marked INSERT. You'll notice that everything from the cursor on disappears—but only temporarily. ADAM has given you space to type your insertion while keeping track of what will follow (the original end of your sentence). Third, type the words "so long," or any insertion of your own. You can continue typing an insertion as long as you wish because ADAM has given you the equivalent of pages of blank space for making additions.

Once you've finished typing your insertion, you'll want to reconstruct your old sentence with the insertion in place. Tell ADAM you're done by pressing Smart Key VI, labeled DONE. Exactly as you had hoped, your entire sentence reappears on the screen with the new words added: "It's been a hard day's night, and I've been working so long like a dog." Congratulations! Not only have you learned to insert text with SmartWriter, you've proved yourself to be a virtuoso songwriter as well.

Clearing the Screen and Work Space

When you use the delete function, you can remove only one screenful of text at a time. You can delete all of the text on the screen more easily and quickly by pressing the CLEAR command key. You won't need to highlight anything. Simply press Smart Key V, labeled CLEAR SCREEN. ADAM will ask you to press Smart Key VI to confirm your command. This is a safety feature for the protection of your work. If you want to erase the entire document from ADAM's work space memory, press Smart Key VI, CLEAR WK-SPACE, rather than Smart Key V after you push the CLEAR command key. This will not affect anything that you have previously stored on a digital data pack.

Printing With ADAM's Daisy Wheel Printer

Suppose you're ready to print your finished text. You've learned to enter and edit a document on ADAM's screen, to store your work on one of ADAM's digital data packs, and to retrieve your text from memory. Now it's time to make a "hard copy" of your text—on paper—to bring all of your hard work to fruition.

With SmartWriter, it's possible to print text virtually anytime you wish—either before you correct errors or after. You also have the choice of printing an entire document or only what's on the screen. Finally, by using fanfold paper made from numerous connected sheets, it's possible to print multiple copies of your text without constantly feeding paper into your printer. This way, while ADAM is busy printing, you are free to do other chores.

To begin printing, press the command labeled PRINT on the right side of ADAM's keyboard. You'll notice that the labels on the bottom of your screen change. At this point, ADAM wants to know what you want to print—only what's on the screen, or the contents of your entire document (the work space). Press IV to print a screenful of text, or V to print the whole work space.

Once again the Smart Key labels will change. This time ADAM asks you a series of questions. By pressing Smart Key II, for example, you tell ADAM if you are using single sheets of paper or fanfold connected sheets. By pressing Smart Key III, you tell ADAM the page number of your first page of text. Try pressing key III and watch what happens. The page number on your screen, which had been 1, should now begin increasing. Keep pressing this key until you reach the page number that you'd like ADAM to print first. If you don't want to worry about pagination at all, press Smart Key IV, labeled NO PAGE #. This shuts off ADAM's paging function.

You've now told ADAM everything it needs to know to print your text. At this point the only thing remaining to do is to insert paper into ADAM's printer—either a single sheet or fanfold, whichever you've specified.

Insert the paper by pulling forward the paper-release lever on the left side of ADAM's printer; then slide the paper under the back of the roller. Once the paper is centered, you can lock it in place by returning the release lever to its original position. Now roll the paper forward, exactly as you would on a regular typewriter, by turning the platen knob on the right side of the printer. Keep rolling the paper forward until the top is just under the paper bar.

You're ready to print. Press Smart Key V, labeled PRINT. Almost instantly, ADAM's printer comes to life. If everything is working well, the

daisy wheel should now begin printing. Not only will your final printed copy be of letter quality, exactly as if you had typed it yourself, but it will be printed at 120 words per minute.

Margins, Tabs and Other Potpourri

If you're using narrow paper, envelopes or mailing labels in ADAM's printer, you'll probably want to reset the margins before you begin printing. ADAM will restructure all of the text in the document to fit within the left and right margins you set, and will also vary the page breaks in the text to adjust for the top and bottom margins.

Begin by pressing Smart Key I, MARGIN/TAB/ETC; then select the type of margin that you want to change. Assume that you need a left margin of 10 spaces, a right margin of 50, a top margin of 2 lines, and a bottom margin of 20. Press Smart Key II, HORIZ MARGIN. You will then see the current left and right margins, which will be set at 10 and 70 if you've just turned ADAM on. The left margin is precisely where you want it. To change the right margin, press Smart Key IV, RIGHT 70. Nothing happens except for a sound from the television's speaker. But something has happened. You can now use the left and right cursor controls to change the number 70 to any value from 80 down to the value of the left margin. Hold down the left arrow until the number changes to 50. If you overshoot, just release the key and press the right arrow. You can now press Smart Key V in order to change the top and bottom margins. When the Smart Key labels change, you simply tell ADAM which margin to change by selecting the appropriate Smart Key and using the same arrow keys to alter the numbers listed on the screen. Push Smart Key VI to tell ADAM that you've finished and to return to the original set of Smart Key labels.

To set and remove tabs for various indentations, begin by pressing Smart Key I, MARGIN/TAB/ETC. Next press Smart Key IV, labeled TAB. Use the left and right cursor-control keys to move the cursor to the points at which you want to set tabs or remove existing tabs with Smart Keys III and IV. Notice that ADAM has several preset tab positions. You can remove these if you wish. Carefully watch the horizontal scale at the top of the screen to see exactly where you are on the page. Pressing Smart Key V will erase all of the tabs that have been set. Smart Key VI again tells ADAM that you have finished.

You can also tell ADAM the size of paper you plan to use by pressing Smart Key I again. This will let you choose between letter size and legal size paper.

If you press Smart Key V, LINE SPACING, after pressing Smart Key I, MARGIN/TAB/ETC, you will see three labels—UP, DOWN and DONE. In addition, you will see 1 (the number one) in the lower left region. This number is the current line spacing. ADAM will print single space unless you insert carriage returns between the lines. Pressing Smart Key IV twice will change the 1 to 1½ and then to 2. ADAM will then print double space. You can vary the spacing in increments of ½ from 1 to over 50. It won't affect the way the text looks on the screen; it will affect only the printed copy.

Another label that appears after pressing Smart Key I corresponds to Smart Key VI and is END PAGE. This causes a small character, a light colored "E" in a dark square, to appear at the cursor's location in the text. When ADAM sees this character while printing, it knows that the end of the page has arrived and that the next word should appear at the beginning of the next page. You can use the backspace or delete keys to remove this character, just like any usual character.

Screen Options

When you press Smart Key II, you will have the option of turning off the sound—the beeps ADAM makes when you press a key—either partially or fully. The PARTIAL SOUND option (Smart Key IV) turns off all sound except when a Smart Key is pressed. FULL SOUND (Smart Key V) is ADAM's normal mode when first turned on. NO SOUND (Smart Key III) is self-explanatory.

Smart Key II offers you COLOR SELECT, which will provide you with several options pertaining to the color of the screen and the color of the text.

Smart Key VI, labeled MOVING WINDOW, will cause the black roller at the bottom of ADAM's screen to disappear. The text will now be arranged like a printed copy, with the exception of the line spacing. It is called a moving window because a line of text can extend beyond the screen where it can't be seen. You can use the cursor controls to move along the line of text and, as new words appear on one end of the line, words disappear on the other. It is as if you are physically moving a window on a wall in order to see something behind it. While you are in the

moving window mode, try this experiment: Set the left margin to 1 and the right margin to 80. Move the cursor over several lines of text that you don't particularly want to keep, or press the key marked HOME. Then try to print the text. On some machines, all but the first line of text will be destroyed. This is what computer users call a *bug*—a glitch in the program.

Getting Out of Trouble

ADAM has two keys that might save you trouble someday: <ESCAPE> and <UNDO>. These keys cancel commands previously given. <ESCAPE> is used to interrupt. It must be used before pressing a Smart Key labeled DONE in a sequence of commands. <UNDO>, on the other hand, actually reverses the effects of several commands even after they've been executed. It must, however, be pressed immediately after the command, before pressing another key. <UNDO> undoes the commands that remove text—BACKSPACE, DELETE and CLEAR. The UNDO and ESCAPE keys can lengthen the useful life of your ADAM, since computers that have been thrown across the room don't usually function for very long.

CHAPTER 4
ADVANCED WORD PROCESSING

Moving Text

One of the most important features of any word processing system is the ability to move or rearrange blocks of text. ADAM can quickly perform this task; you need only give the proper commands.

To start, choose any text that you want to rearrange. For example, suppose you typed the following:

> Violets are blue.
> Roses are red.

Now suppose you'd like to reverse the position of these sentences so that they read:

> Roses are red.
> Violets are blue.

The only way you can achieve this on a conventional typewriter is by erasing at least part of what you've typed or by retyping the entire page. But with SmartWriter, it's possible to rearrange text quickly and neatly on your screen before anything is printed on paper. Using the MOVE/COPY command key, you can instruct ADAM to erase a sentence from the screen, store it in memory, and reposition it at any other point in your text. What follows is a list of the steps needed to do this.

1. Press the command key MOVE/COPY at the upper right of ADAM's keyboard. This gives a new set of Smart Key labels.
2. Press Smart Key V, labeled MOVE. The Smart Key labels will change again.
3. Using the cursor-control keys, move the cursor until it is under the first character of the block of text you wish to move. In this case, we'll be moving the sentence "Violets are blue," so move the cursor to the "V" in "Violets."
4. Press Smart Key IV, labeled HI-LITE FIRST. This will underline, or highlight, the "V." If you make a mistake and accidentally highlight the wrong character, you can press Smart Key V, labeled HI-LITE ERASE, and the letter will no longer be underlined.

5. Move the cursor to the last character of the block to be moved. Be sure that you can still see the character you highlighted in step 4. If the block of text is so large that you can't see the beginning and end of the block at the same time, you'll have to break the block into smaller pieces and move each piece separately. In our example, this would mean placing the cursor under the triangular symbol (the carriage return character) following the word "blue."

6. Press Smart Key V, labeled HI-LITE LAST. Notice that all of the text between the two highlighted characters disappears from your screen. The block of text has been stored in ADAM's memory, awaiting your instructions indicating where you want the text to reappear. Pressing ESCAPE/WP at this point has no effect.

7. Move the cursor to the position where you want the beginning of the stored text to appear. This time, put the cursor one line below the "R" in "Roses."

8. Press Smart Key V, MOVE, to make the text reappear. When it returns, it should look exactly as it did formerly, except that the entire block will be located at its new position, as follows:

Roses are red.
Violets are blue.

With practice, you will be able to rearrange paragraphs, sentences and phrases in a matter of seconds. This procedure can also be used to improve the organization of your writing as you reorganize your thoughts.

Copying Text

You noticed that when you moved the sentence in the preceding example, it disappeared from its original location. Suppose that you are particularly fond of violets and that you did not want the sentence "Violets are blue" to disappear. That is, instead of erasing and moving the sentence, you wanted to copy it and place the copy elsewhere. One result might look like this:

Violets are blue.
Roses are red.
Violets are blue.

Copying text follows a procedure similar but not identical to that for moving text. Here are the steps involved:

1. Press the command key MOVE/COPY at the upper right of ADAM's keyboard. The Smart Key labels will change.

2. Press Smart Key VI, labeled COPY. The Smart Key labels will change again.

3. Using the cursor-control keys, move the cursor until it is under the first character of the block of text you wish to copy. In this case, we'll be copying "Violets are blue," so move the cursor to "V" in "Violets."

4. Press Smart Key IV, labeled HI-LITE FIRST. This will underline, or highlight, the "V." If you make a mistake and accidentally highlight the wrong character, you can press Smart Key V, labeled HI-LITE ERASE, and the letter will no longer be underlined.

5. Move the cursor to the last character of the block to be copied. Be sure that you can still see the character you highlighted in step 4. If the block of text is so large that you can't see the beginning and end of the block at the same time, you'll have to break the block into smaller pieces and move each piece separately. Again, this would mean placing the cursor under the triangular symbol (the carriage return character) following the word "blue."

6. Press Smart Key V, labeled HI-LITE LAST. In doing so, you etch the block of text to be copied into ADAM's memory.

7. Move the cursor to the position where you want the beginning of the block of text to be duplicated. This time, put the cursor one line below the "R" in "Roses."

8. Finally, press Smart Key V, COPY, to copy the text and make it reappear where the cursor resides. When ADAM has finished, the text should look exactly as it did formerly, except that another sentence, identical to the first one, now appears under "Roses are red." You now have three sentences—the original two and a copy, as follows:

Violets are blue.
Roses are red.
Violets are blue.

Suppose that you not only like violets, but are absolutely infatuated with them. Repeat steps 7 and 8 several times and you will get something like this:

Violets are blue.
Roses are red.
Violets are blue.
Violets are blue.
Violets are blue.
Violets are blue.
Violets are blue.

When you've finished making violets, you simply proceed by going on to step 9.

9. Press Smart Key VI, DONE. The original Smart Key labels should reappear at the bottom of the screen.

It is important to remember that any highlighting done before you press <MOVE/COPY> will disappear, which is not how <DELETE> works. The characters will still be there, but the colored highlighting line beneath the text will no longer be present.

Searching for Text

Imagine that you are nearly finished typing a 20-page paper on the advantages of having your own computer. You remember that somewhere in your text you used the phrase "roses are red" as an illustration of one of your points. All of a sudden you decide that you want to review your example. You are now faced with the problem of scrolling back through your document, page by page, until you find the phrase. Why not let ADAM do the tedious work for you?

Notice that the function corresponding to Smart Key III is labeled SEARCH. This command tells ADAM to locate any character, word or phrase (up to a maximum of 32 characters). ADAM will do your searching if you follow these simple steps:

1. Move the cursor to the beginning of the document or page that you want ADAM to inspect. Always remember that ADAM searches from the cursor position to the end of the document. For example, if you were to place the cursor at the beginning of page 7, ADAM would ignore pages 1 through 6 and search from the beginning of page 7 to the end of the text.

2. Press Smart Key III, labeled SEARCH. The labels will change and ADAM will ask you to type in the word or phrase for which you are looking.

3. Type in the character, word or phrase to be found. In this case, you could type "roses are red," "rose," "red," or even "RED" (without the quotation marks or commas). Type "are red" for now. Notice that ADAM ignores differences in capitalization when searching. Also, ADAM accepts everything you tell it quite literally. If you told ADAM to search for all occurrences of the word "red," ADAM would find "red" even when it appears as part of another word, as in "redwood," "redhead," or "scared." A trick often used to prevent this from happening is to precede the word for which ADAM is searching by a space. Typing " red" instead of "red" will prevent ADAM from locating the "red" in "scared" but,

unfortunately, not in "redwood." Typing " red." would locate all occurrences of the word "red" that are followed by a period. Be as specific as possible; for instance, "roses are red" will probably not occur as part of any other word or phrase.

4. Press Smart Key VI, which is labeled START SEARCH. The cursor will move to the first occurrence of the phrase "are red" and the labels will again change. If ADAM has located a phrase other than the particular one you had in mind, press Smart Key IV (labeled SEARCH NEXT) until ADAM finds the phrase.

5. Press Smart Key VI, DONE. The original Smart Key labels will reappear, and you can begin editing your text once more.

Replacing Text

Suppose that you remember that you'd promised to write a long letter to your friend Carla. You have only a few minutes before the postman will arrive at the mailbox around the corner. Realizing that you'd written to your friend Sheila the day before and had saved the letter, you decide that your only chance to meet the deadline is to use that letter and change the name wherever it occurs. Or suppose, given our old "roses" and "violets" example, you decide that roses should be white rather than red. ADAM can easily solve either task if you follow these instructions:

1. Move the cursor to the beginning of the document or page that you want ADAM to inspect.

2. Press Smart Key III, labeled SEARCH. The labels will change, and ADAM will ask you to type in the word or phrase for which you are searching.

3. Type in the character, word or phrase to be found. For now, just type "are red."

4. Press Smart Key VI, which is labeled START SEARCH. The cursor will move to the first occurrence of the phrase "are red" and the Smart Key labels will again change.

5. At this point you have three choices: (a) You can tell ADAM to replace every occurrence of the phrase "are red" from this point to the end of the document automatically; (b) you can decide to replace the phrase that ADAM has just found for you and retain the option to control the replacement of each subsequent occurrence of the phrase for which ADAM is searching; or (c) you can command ADAM to leave this occurrence unaltered and go on to the next one, again with the option of changing subsequent occurrences.

5a. To replace this and all subsequent occurrences of the phrase automatically, press Smart Key VI (labeled REPLACE ALL). ADAM will then ask you to provide the replacement text. In this case, type "are white." ADAM will replace every subsequent occurrence of "are red" with "are white," and then return to the original Smart Key labels.

5b. By pressing Smart Key V, labeled REPLACE, you are telling ADAM to replace only this occurrence of the phrase "are red." If you choose this option, you can't choose the REPLACE ALL option unless you start from scratch. You will, however, retain the choice of replacing or not replacing all subsequent occurrences.

5c. If you want to leave this occurrence unchanged and tell ADAM to find the next one, press Smart Key IV, SEARCH NEXT. ADAM will search for the next occurrence of the phrase and again provide you with the option of changing it or leaving it unaltered.

6. When you've finished making your desired replacements in the text, press Smart Key VI, DONE. You are then returned to the original Smart Key labels. Many times you may find yourself typing the same long phrase or word again and again in a document. For example, you may be typing a paper on Greek literature and often refer to the writer Aristophanes. A shortcut you might like to remember is to type an abbreviation throughout the document. Let's use the letters "AAA" in this case. When you finish, go back to the beginning and tell ADAM to search for each occurrence of your abbreviation, AAA in this instance, and to replace it with the desired word, Aristophanes. You'll find that this little trick can really save you some time. To avoid confusing ADAM, try to use odd combinations of letters that don't ordinarily occur in the language, such as one letter appearing three times as in this example.

Superscripts and Subscripts

Having successfully completed your paper on Greek literature, you decide that it's time to complete a paper on your science project. You realize that in this paper you will need to type several formulas, such as H_2SO_4 or $e = mc^2$. On a typewriter, you must shift and roll the paper in order to enter the characters that appear above and below the normal line of type. The 2 in $e = mc^2$, which appears above the rest of the script, is called a superscript. Superscripts are often used in reports to refer to footnotes. The 2 and the 4 in H_2SO_4 are called subscripts since they lie below the rest of the script. Fortunately, you can tell ADAM to print a superscript or subscript and forget about the headaches associated with moving the paper up and down or back and forth.

You can do this very easily by following these instructions:

1. Type until you come to the point where you want a superscript or a subscript to appear. For example, type the e = mc of $e = mc^2$.

2. Press Smart Key VI, labeled SUPER/SUBSCRIPT. The labels will change.

3. Press Smart Key VI, labeled SUPERSCRIPT, if you want the following characters above the line, or Smart Key V, SUBSCRIPT, if you want the characters that follow to appear below the line. The labels will change again. A special superscript character (an upside down L facing right) or subscript character (an L shape facing right) will appear on the screen in the text.

4. Type in the characters that you want to be superscripted or subscripted at this place in the text. This, for example, could be the 2 in $e = mc^2$.

5. Press Smart Key VI, DONE. The original labels will reappear at the bottom of your screen, and another special character will appear on your screen just after the text that you superscripted or subscripted. The ending superscript character resembles an upside down L facing left, and the ending subscript character looks similar to a left-facing or backward L.

In the above examples, you never see the sub- or superscripted characters on your screen. All you see are the special symbols for superscript and subscript. But these symbols act as messages to your printer to do the necessary subscripting or superscripting. And they're messages to you as well that the writing below and above the line is being done.

One final point. Suppose that you had originally just typed "e = mc" and later, after typing other words beyond the equation, decided to add the superscript, 2. It's no longer possible simply to return to your equation and follow the regular superscript procedure outlined above. Instead, you'll have to insert the superscript using the insert procedure. So start the insert routine by pressing the INSERT key. When ADAM opens space for your insertion, follow steps 2 through 5 above. Then press Smart Key VI, DONE. The characters you typed should be inserted in the proper place. It's smooth sailing from here on. You are now an expert with SmartWriter.

CHAPTER 5
BEGINNING WITH SmartBASIC

In an ideal world, computers like ADAM would speak English. We would use the same words to talk to ADAM that we use to converse with family and friends. If, for example, we wanted ADAM to count from 1 to 100, we would say precisely that—"Count to 100, you extraordinary machine!"—and ADAM would respond in kind. Similarly, if ADAM wanted to send a message to us, it too could use normal speech, saying things like "That's quite a hefty assignment you've given me," or "I'm working on your problem." Computing would be that easy if ADAM spoke English.

Unfortunately, this ideal computer world is not yet upon us. Although enormous strides have been made in the area that scientists call artificial intelligence, which would among other things enable computers to speak, understand and interpret regular English sentences—the fact remains that for most computers, including ADAM, English is far too complex a language to use. Not only are there millions of words, but there are equally many nuances—shades of meaning—that ADAM would have to understand to speak English. Even if we could teach ADAM the varied meanings of millions of English words, it's unclear if ADAM could learn English grammar. After all, many foreigners have trouble learning it. Imagine how difficult it would be for a machine like ADAM to learn!

This chapter is about another type of language, one so precise and easy to use that both you and ADAM can understand it. Indeed, the language we'll be discussing was designed especially for computers like ADAM more than thirty years ago by scientists at Dartmouth College in New Hampshire. They concluded that everyday English simply wouldn't suffice for the computing machines of the time—English was simply too complex. Instead of using everyday English to talk with computers, the Dartmouth scientists designed a language that used many familiar English words and sentences but lacked the subtleties or embellishments that would confuse a computer. That first computer language, called BASIC, has been honed to perfection, and nearly all computers understand it. The version of BASIC that ADAM speaks is called SmartBASIC. Although SmartBASIC may seem foreign and cryptic to you at first—as would any new language—a little further description should dispel any concerns you have.

For starters, SmartBASIC uses many words you already know. For example, if you want SmartBASIC to print, you use the command PRINT. Similarly, if you want SmartBASIC to skip to another location as it processes information, you use the command "GO TO." To have ADAM read a list of numbers, use READ, and if you want to input those same numbers from ADAM's keyboard, use INPUT. These and other SmartBASIC commands will be discussed in detail later in the book. For now, though, take pleasure in the fact that most of the commands in SmartBASIC sound and act as you'd expect.

This, however, is not the main feature that distinguishes SmartBASIC from other versions of BASIC. Rather, what sets SmartBASIC apart is that it will tell you if you have made an error in your work, and it will inform you of your mistake immediately, before you have worked for hours on a particular task. If, for example, you misspell a command (and therefore ADAM does not recognize it), SmartBASIC will tell you at once where and how you've blundered. SmartBASIC may even suggest ways to fix your error. The language thus earns the name SmartBASIC, in recognition of its ability to pinpoint errors early on and to suggest ways of improving. This is in contrast to many other versions of BASIC that wait until you're done with all your work before spotting an error.

Being a language, SmartBASIC also has a grammar. Just as English grammar lays out the rules and procedures for sentence structure and word usage, SmartBASIC's grammar sets the rules for writing computer programs. A program is simply a set of instructions in SmartBASIC that tells ADAM what to do.

All this is not to say that SmartBASIC is the same as English. It's not. Certainly you will never hear people speaking SmartBASIC. And you will never see SmartBASIC written in a newspaper or magazine. But you will see SmartBASIC (or another version of the BASIC programming language) being used widely by beginning and advanced computer users to communicate with their machines.

While many of the words and phrases used in SmartBASIC are familiar, there are also far fewer phrases than there are in English. The language, in short, has been shrunk. There are no adjectives or adverbs, no modifiers or other embellishments that ordinarily enrich a language. Instead, SmartBASIC consists of only about a hundred choice words. Precise, direct and unequivocal, these words, called commands, do nothing but tell ADAM what to do. Indeed, if there is a critical difference between everyday English and programming in SmartBASIC it is this: While everyday English relies on subtleties and other shades of meaning to convey a thought, in SmartBASIC the emphasis is on precision. Every word in a SmartBASIC program has a specific definition and always

means the same thing. Each SmartBASIC phrase also retains a meaning, no matter how it is used in a program. One could say that English deals in greys, SmartBASIC in black and white. English is mutable and subjective, SmartBASIC fixed and objective.

Some people, of course, take issue with having to speak to ADAM in such a rigid tongue. Why, they ask, should a language like SmartBASIC have such hard-and-fast rules? And why no adjectives or adverbs? The answer is that it could not be otherwise. Since ADAM is a machine made by man and cannot think for itself, it cannot be asked to interpret the English language as we know it. The best we can hope for is the ability to talk with ADAM in a clear, direct, unambiguous way that ADAM understands, even if it means using a language far less sophisticated than English. We think SmartBASIC fits this bill well.

In fact, we believe that you'll find SmartBASIC to be an elegant, descriptive language—despite what others have called SmartBASIC's simple "lockstep ways." SmartBASIC can be enormously powerful in uses, ranging from balancing your checkbook or keeping track of names and addresses to performing complicated mathematics. All it takes to use SmartBASIC is patience and a willingness to learn.

Consider yourself to be studying a new language—in this case, a computer language. As you add new SmartBASIC words and phrases to your "vocabulary," try writing them down to remember them easily. Better still, try practicing your new skills by using SmartBASIC phrases in programs, just as you would practice new words if you were learning a new language, such as French or Spanish. Your goal should be to become fluent in SmartBASIC—to learn all its words and phrases. It's only when you know SmartBASIC well that you can speak to ADAM effectively and get the most out of its powerful electronic mind.

Getting Started

If you had no trouble beginning work with SmartWriter, chances are you will find getting started with SmartBASIC easy, too. To begin, turn ADAM on, place the SmartBASIC digital data pack into ADAM's tape drive and press the reset button on top of the memory console. The drive will whir as the tape advances. This means ADAM is loading SmartBASIC into its electronic brain so that it can accept your commands. You'll know ADAM is ready to receive your instructions when a bracket (]) appears on the screen. You're now ready to utter your first words in SmartBASIC.

Begin by typing the word NEW followed by the carriage return key. The NEW command tells ADAM you are about to input a new set of instructions—a new program. Indeed, whenever you want to begin a new program, simply type NEW and the bracket will reappear on your screen. You can even type NEW in the middle of a program, and ADAM will disregard what you've already written and begin again.

Since computers were created to solve difficult math problems at fantastic speeds, it seems only appropriate that our first program should involve numbers. Let's get started by writing a simple program to add three numbers: 457, 821 and 1021000. After typing NEW, try inputing the following (don't type the bracket (]), it should already appear on your screen):

```
]new
]10 print 457+821+1021000
]20 end
```

This program contains two separate statements, and each statement directs ADAM to do a specific job. The first line, labeled 10, consists of SmartBASIC's PRINT command followed by the three numbers we wish to add. The PRINT command simply tells ADAM to print on the screen whatever follows—in our example, the sum of three numbers. So when ADAM encounters line 10, it thinks: "Take the numbers 457, 821, and 1021000, add them together, then display the result on the screen." Since ADAM knows how to recognize numbers and perform arithmetic, it has no problem doing the calculating job. Once ADAM is done computing, it proceeds to line 20, where the END command indicates that it has reached the end of your program. It's that simple.

If you try typing in this program on ADAM's keyboard, however, you'll find that ADAM will sit there listlessly once you're done. You can bang on the keys, plead and cajole, but ADAM will stubbornly refuse to give you the answer you seek. Is there anything wrong? Probably not. Is ADAM lazy? That's probably untrue also. The problem is that you haven't told ADAM to *execute* your program—to read through it and follow each instruction line by line. What you need is a way to get ADAM started.

In SmartBASIC this is achieved through the RUN command. RUN tells ADAM to read through your program's instructions in order and follow them. Try typing RUN, followed by a carriage return, after you've typed in our model program. If all goes well, a solution to our equation should appear on the screen:

```
1022278
```

Sure enough, 1022278 is the number you're looking for. It's the sum of our original numbers: 457, 821, and 1,021,000. Not only has ADAM added these numbers as requested and printed them on your screen, it has done so at breakneck pace—certainly faster than you could add the numbers by punching them into a pocket calculator. Later, we'll be doing far more complex calculations with SmartBASIC, and you'll see that ADAM still works amazingly swiftly when the jobs get big.

What are the lessons that you should take away from this first program? For starters, you should have an elementary understanding of ADAM's PRINT command. Just to review, PRINT tells ADAM to print on the screen whatever follows—in our example, the sum of three numbers. Since PRINT is one of the most widely used commands in SmartBASIC, we think it's worth learning early on. You should also remember that the END command in SmartBASIC informs ADAM it has reached the end of a program. And by typing RUN once you've finished writing a program, you tell ADAM to execute your program's instructions.

You also have the option of giving ADAM commands one at a time and getting ADAM's response without delay and without typing RUN. For instance, type:

```
]print 457+821+1021000
```

ADAM responds with:

```
1022278
```

Since ADAM responds immediately to your command, it is said to be working in *immediate mode*. No line numbers or RUN command are necessary. The major disadvantage of the immediate mode is ADAM's inability to save those commands for future use. The immediate mode is fine for using ADAM like a big calculator, but it doesn't really take advantage of ADAM's full potential.

When you typed the numbers at the beginning of the lines, you were telling ADAM to wait until you gave the RUN command before following your instructions. This is called *program mode,* since a list of instructions with line numbers constitutes a program for ADAM.

You'll notice, too, that every line in our SmartBASIC program begins with a number—in our example, the numbers are 10 and 20. These serve a dual purpose. First, they signal to ADAM that you are starting a new line. But more important, it's only by reading the numbers at the beginning of each line that ADAM knows the order in which to follow your instructions. ADAM executes the line with the lowest number first. It then proceeds to follow the command with the next highest number. In our example above, ADAM would execute line 10 first, then line 20, regardless of the order in which the lines were typed.

Let's get a fresh start here. Type NEW to erase any old programs, and then type HOME. The HOME command erases the screen without affecting any of the programs.

Suppose you wrote a program based on one of the most inspiring lines from John F. Kennedy's inaugural address. If you wrote a program like the one below, you might think you would confuse ADAM, since it looks as if you have written the late President's hallowed words in reverse:

```
]20 print "Ask what you can do
for your country."

]10 print "Ask not what your co
untry can do for you"

]30 end
```

However, ADAM wouldn't be confused at all. Even though ADAM wasn't around to hear the President's address, he still is able to read the lines in numbered order, beginning with line 10 and then moving on to line 20. ADAM would in effect rearrange the program, thinking of it exactly as President Kennedy spoke it, as follows:

```
]10 print "Ask not what your co
untry can do for you"

]20 print "Ask what you can do
for your country."

]30 end
```

Notice that when you get to the right edge of the screen, ADAM displays the letters that you've typed on the following line. Even though the word "country" appears to be broken, it really isn't. It's only a cosmetic flaw caused by the lack of anything akin to word wrap in SmartBASIC. It can be fixed by using several shorter PRINT statements.

Just to repeat: Whenever you write a SmartBASIC program, each line must have a line number. These tell ADAM the order in which to read instructions. ADAM reads the line with the lowest number first, then proceeds to the lines with higher numbers. Although you can use any whole numbers to order a SmartBASIC program, it's probably best to use multiples of 10, just as we did above. A brief example should illustrate why this is a good idea. Suppose you've written the following program:

```
]new
]10 print 724+958+535
]20 end
```

If you type RUN, ADAM will immediately put its wondrous electronic mind to work and deliver the answer you seek: 2217. But suppose you wanted ADAM to perform another calculation before ending—to add, say, 345,1236 and 627. In SmartBASIC, you could do this easily and neatly by writing a separate instruction within the same program, numbered 15.

In other words, despite the fact that you've finished typing your original program, you could still squeeze in a line to perform the second calculation you want. After the end of your original program, simply type:

```
]15 print 345+1236+627
```

Although line 15 is standing by itself and may appear not to be part of your program, it is. To see that this is the case, try typing the word LIST followed by a carriage return. The LIST command tells ADAM to list out, in numbered order, the individual lines of your program. If all goes well, ADAM should print on the screen the following:

```
]10 print 724+958+535
]15 print 345+1236+627
]20 end
```

If you tell ADAM to run, you'll get two numeric answers, exactly as the program requests. Your output on the sceen should be as follows:

	not displayed
2217	(The sum of 724, 958, and 535)
2208	(The sum of 345, 1236, and 627)

There is, finally, one other lesson we think you should garner from this primer on programming in SmartBASIC. You'll notice that in all our programs every line ends with a carriage return. Although there are certain exceptions to this rule (which we'll tell you about), it's generally true that all lines in SmartBASIC must end with a return, regardless of the command on the line. By typing a return, you tell ADAM that you have finished an instruction and are ready to begin a fresh one. In fact, ADAM, being a stubborn creature often set in its ways, won't read what you've typed until you hit a carriage return. Unless you type a return, it's as if you hadn't typed a line at all. It's important, therefore, to type a return each time you wish to begin a new line, and then to start that new line with a fresh line number.

Arithmetic Calculations

In your first program, you learned to add numbers using SmartBASIC's PRINT command. As part of this, you learned the meaning of the LIST, END, NEW, and RUN commands. It turns out that performing any of the other arithmetic functions in SmartBASIC can be equally easy, provided you learn some simple conventions.

Suppose, for example, you wanted to subtract one number from another. Consider the following program:

```
]10 print 7231 - 4326
]20 end
```

After you type RUN, ADAM should respond by computing the difference: 2905. In other words, subtraction is performed exactly as you would do it with pencil and paper, using a minus sign (which appears on ADAM's keyboard just above the P). ADAM even understands negative numbers. Try the following program:

```
]10 print 4305 - 7462
]20 end
```

When you type RUN, ADAM should answer with a negative solution: −3297. Although there are limits to the size of numbers ADAM can understand, for most uses both negative and positive solutions can get as large as need be.

Now suppose you'd like to multiply numbers using SmartBASIC. Ordinarily, you'd use a conventional multiplication sign—x—to find a product. But not with SmartBASIC. Instead of an x you use an asterisk (*) to indicate multiplication, like this:

```
]10 print 2*4*6
]20 end
```

After you tell ADAM to run the program, the product, 48, should appear on your screen. Virtually the same procedure is used for division, except that you use a slash (/) to tell ADAM you'd like to divide. The slash key appears in the same place on ADAM's keyboard as the question mark. A program involving division would look like this:

```
]10 print 3432/8
]20 end
```

After you type RUN, ADAM will again offer up a solution, and as usual in a flash. This time the answer is 429.

Although the procedures for using SmartBASIC for arithmetic vary depending upon whether you're doing addition, subtraction, multiplication or division, there are also some common rules that apply regardless of the operation.

For example, you've probably been mystified by the fact that none of the numbers in our examples contain commas, even very large numbers. That's because ADAM does not understand commas in numbers. The following table gives some examples of how numbers should be typed in SmartBASIC:

Number	SmartBASIC Version
3,000	3000
1,234,567	1234567
2,000,000	2000000

It's also important that you know something about scientific notation—a convenient way of expressing extremely large or small numbers in the small space of a computer screen. The reason scientific notation is important is that ADAM uses scientific notation when numbers are too big to fit on the screen. Although this happens only rarely, you'll know ADAM has reached this point and is using scientific notation when numbers include an E. For example, suppose ADAM says the answer to a problem is:

7.52345E9.

The "E9" means that you should move the decimal point 9 spaces to the right. Instead of being about 7.5, the number listed above is really 7523450000. Similarly, an E followed by a negative sign would mean move the decimal point to the left. For example, 8.546E−3 means .008546, and 4.732E−5 translates into .00004732. It's that simple.

Finally, since you already know that ADAM reads program lines in numbered order, you'll want to know the order in which ADAM conducts arithmetical operations within a specific line. If, for example, an equation contains a number of operations—say, addition, subtraction and multiplication—which will ADAM perform first? Fortunately, scientists at Coleco have designed ADAM so that it follows essentially the same rules that schoolchildren learn. The procedure goes like this:

First ADAM looks for parentheses. It performs operations inside the parentheses before anything else.

Next ADAM performs all multiplication and division, always working from left to right.

Finally ADAM performs addition and subtraction, again working left to right.

Although all this may seem cryptic at first, a few examples should make things clear and put you perfectly at ease:

Suppose you write a program like this:

```
]10 print 6*7+4/2+2
]20 print 4/2*5+2-7
]30 end
```

Many people would be baffled trying to solve such twisted equations, but not ADAM. When you tell ADAM to run this program, it will start by computing the value of the equation in line 10. Using the rules stated above, it will look first for parentheses. Finding none, ADAM will then perform multiplication and division, left to right, by computing the value of 6*7 and 4/2. The last step will be to add the results together. It will add 42 to 2, and then it will add 2 again. The final result will be 46.

In the equation on line 20, ADAM will again start by seeking parentheses. Finding none, it proceeds to multiplication and division. It will divide 2 into 4, then multiply the result by 5 to get 10. Having completed multiplication and division, ADAM can now proceed to the lower-priority operations—addition and subtraction. To 10 ADAM will add 2, then subtract 7, exactly as the equation says it should. The final result will be 5.

One easy way to think about all this is to imagine you have placed invisible parentheses in the equations above, following all the rules for order of operations. In fact, from ADAM's view, the parentheses are there; they just haven't been typed in.

You can, therefore, think of line 10 not as a confusing tangle of numbers and mathematical symbols, but as an ordered, easy-to-understand equation with parentheses—in short, the type of equation to which you're accustomed. Just make believe the parentheses were there.

Instead of thinking of line 10 like this:

```
]10 print 6*7+4/2+2
```

You can think of it like this:

```
]10 print (6*7)+(4/2)+2
```

And you can consider line 20 to be equivalent to the following:

```
]20 print (4/2*5)+2-7.
```

By imagining these fictitious parentheses, you can understand the "thinking" that ADAM goes through when it executes your programs. You

understand the logic or method that ADAM follows, the computer's precise order of operations when it comes to arithmetic. Reason emerges from the chaos.

Displaying Your Results

So far you've learned about using ADAM as a high-priced calculator, albeit an extremely fast one. Working with SmartBASIC's PRINT command, you are now able to have ADAM perform virtually any mathematical computation you desire, whether it involves addition, subtraction, multiplication or division. Knowing the rules that ADAM follows for order of operations, you can calculate answers even to very complicated problems.

Now let's see if ADAM can understand words. Suppose you'd like ADAM to print a line from the Broadway hit musical *Fiddler on the Roof,* as follows:

"Matchmaker, matchmaker make me a match. Find me a find. Catch me a catch."

To get ADAM to print text, again use the PRINT command, but with one important difference. Whereas earlier you followed the word PRINT with the numbers you wanted to perform calculations on, to print text you must enclose it in quotation marks. Try typing the following program:

```
]new

]10 print "Matchmaker, matchmak
er, make me a match. Find me a
find. Catch me a catch."

]20 end
```

After typing RUN, you won't hear music flowing, but ADAM will belt out the words from the song on your screen:

```
]run

Matchmaker, matchmaker, make me
a match. Find me a find. Catc
 h me a catch.
```

(Notice that the breaks in the words always occur at the ends of the lines, and the breaks are different in the statement you typed than in ADAM's response.)

Now type NEW and try the following program:

```
]10 print "5*7"
]20 end
```

If you type RUN, you should be surprised by your new results. Whereas a similar program would have solved the equation and printed out a result—namely the product of 7 and 5, which is 35—in this case ADAM will print on the screen precisely what's inside the quotation marks. After typing RUN, you should get the following:

```
5*7
```

The reason ADAM does this (rather than compute the product of 5 and 7) is that it considers anything inside quotation marks to be words and prints them exactly as you originally wrote them. Even more surprising, ADAM will print material inside quotation marks without criticism, ignoring errors of grammar, spelling, logic and punctuation. All quoted material is the same to ADAM, sensible or not. The following program illustrates:

```
]10 print "2+2=5"
]20 end
```

Of course, knowing that two and two is four, not five, is kids' stuff. But matters are not that simple for ADAM. What's important from the computer's point of view is that the equation is enclosed by quotation marks. It thus becomes text in ADAM's eyes. The result: ADAM will print the equation on the screen as written. It will display 2+2=5.

There is, however, an answer to this somewhat perverse logic, this twisted arithmetic in which 2+2=5. Although we cannot teach ADAM to fix incorrect numbers or faulty equations inside quotation marks, we can write a program in which all computations appear outside quotes. This has the effect of putting ADAM's electronic brain to work rather than having the computer simply return verbatim what it was given. An illustration shows how this would work. Consider the following program:

```
]10 print "The sum of 2 and 3 i
s " 2+3

]20 print "The difference of 35
and 20 is " 35-20

]30 print "The product of 6 and
7 is " 6*7
```

```
]40 print "The quotient of 100
divided by 10 is " 100/10

]50 end
```

When you instruct ADAM to execute this program by using the RUN command, the answer will be much more to your liking. Beginning with line 10, ADAM will print precisely what's inside the quotation marks. Then it will compute the sum of 2+3 and print the result. You will see the following:

```
The sum of 2 and 3 is 5
```

ADAM will go through the same sort of procedure for lines 20, 30 and 40. It will print:

```
The difference between 35 and 2
0 is 15

The product of 6 and 7 is 42

The quotient of 100 divided by 10 is 10
```

The basic point should now be clear. The PRINT command is one of the most powerful and widely used instructions in SmartBASIC. It can be used to display numbers, words, and any solution that ADAM has found as a result of solving an equation. If you use quotation marks, you're instructing ADAM to return verbatim what you've stored inside them. If you don't use quotation marks, you're telling ADAM that what follows is numerical, and it may call for some calculations.

For now, we'll say that words to be printed on the screen must be inside quotation marks. That way ADAM considers them words and won't try to perform numerical calculations on them. Here is another example.

```
]new

]10 print "For just " 3.98 " you
can buy a machine that in more
expensive stores would cost nea
rly double, or " 3.98*2 "."

]20 end
```

That's quite a mouthful for one PRINT command, but it's perfectly clear to ADAM. After telling ADAM to run the above program, the result you'll get should be as follows:

```
For just 3.98 you can buy a mac
hine that in more expensive sto
res would cost nearly double, o
r 7.96.
```

You'll notice that ADAM printed everything in quotation marks exactly as it appeared. Any necessary calculations, for example, the calculations at the end of the sentence in which 3.98 is multiplied by 2, were performed by ADAM because the numbers involved did not appear within quotation marks. Since we placed the numbers at the end of our sentence, the result of the computation appeared just where we wanted it.

Congratulations! You've mastered an introduction to SmartBASIC. In the next chapter, we'll be expanding your knowledge by describing numerous other commands used in SmartBASIC. Then we'll be offering you some helpful programs for the home, office, or just for fun.

CHAPTER 6
VARIABLES

The problems posed in the last chapter could easily be solved by any five-dollar, four-function calculator. Addition, subtraction, multiplication and division are no longer the challenges they once were before these modern days of calculators and computers. In the next few chapters you will discover how to instruct ADAM to perform tasks that go far beyond the capability of your little pocket calculator.

In addition to being smart and fast, ADAM has a phenomenal memory. ADAM can remember thousands of numbers, names, addresses, words and instructions, which can be used to your advantage.

Suppose you write the following:

```
]X=3
]Y=5
]Z=7
```

Although all three lines of this program are different, they all serve the same function of telling ADAM to assign a particular letter name to a given location in its memory and then to place a number in that location. When ADAM reads the first line, for example, it takes the number 3, carefully places it in a corner of its memory and then identifies that memory location by assigning the letter X to it. We say that X is a variable identifying a unique memory location. And since in this case the memory spot contains a number, we say that X is a numeric variable.

The same sort of reasoning can be used to "translate" the next two lines. In the second line, ADAM is told to take the number 5, place it in memory and assign the letter Y to that spot. Y becomes a variable identifying a memory location with the number 5 in it. And once again, we say that Y is a numeric variable since it identifies a part of memory where a number resides.

You can use ADAM to remember all of the things you're too busy to keep track of yourself. For example, you know that there are 36 black keys on a piano's keyboard. Since you don't want to bother counting them every time someone asks you how many there are, tell ADAM to remember for you. You can do this by typing:

```
]black = 36
```

or

```
]let black = 36
```

The use of LET is optional. Neither including nor deleting it will result in an error. Its use was required many years ago when the original version of BASIC was written, but in modern versions it is unnecessary. LET has probably been kept around just to keep the old-timers happy.

Anytime you want to ask ADAM the number of black keys, tell it to give you the answer by typing:

```
]print black
```

ADAM will respond with:

```
36
```

You can do the same with the white keys. Type in:

```
]white = 52
]print white
```

and ADAM again responds with the correct answer, 52. You can get fancier by now typing:

```
]print "The number of white key
s on a piano is "white" and the number of black keys
is "black"."
```

When you hit the return key, ADAM's response is:

```
The number of white keys on a
piano is 52 and the number of
black keys is 36.
```

To see how smart ADAM really is, a tough question: How many keys are there on a piano's keyboard, both black and white? Simply type:

```
]print white + black
```

and you will get the answer:

```
88
```

Numeric Variables

You have just made a great leap beyond simple calculators, since you have been using common words to represent information. These words are called *variables* because their values can vary or change. They are *numeric* because they represent numbers. Just as regular mailboxes contain labels for easy sorting, ADAM has electronic labels corresponding to the electronic pigeonholes into which it saves and from which it retrieves information. The names you give to the variables are the labels on these pigeonholes.

You could have assigned any value to the variable WHITE. You could even have assigned it the value of another variable, such as BLACK, by typing:

```
]white = black
```

In English, this would be equivalent to saying: "The variable named WHITE assumes the value of the variable BLACK. WHITE and BLACK are both now equal to 36. Note: This is *not* the same as typing:

```
]black = white
```

If you accidentally typed this instead, both variables would now be equal to 52. The variable to the left of the equal sign always assumes the value of the number, variable or equation located on the right. You can easily check this by asking ADAM to print the value of the variables. Since the left side of the equation changes to assume the value of the right side, only a single variable can be located to the left of the equal sign. ADAM will remind you of this if you forget it. For instance, type:

```
]19 = black
```

or

```
]white + black = keys
```

ADAM will chide you by saying:

```
19 = black
   ^
Illegal Command
```

or

```
white + black = keys
      ^
Illegal Command
```

ADAM is providing you with constructive criticism. Not only is it telling you that you made a slight error in the grammar of SmartBASIC, ADAM is also trying to show you where you made the error by placing a caret or arrowhead beneath the location of the error. By putting the arrowhead under the equal sign in "19 = black," ADAM is telling you that after you typed in the number "19," you were then wrong to type an equal sign. You can't put a number in front of an equal sign in SmartBASIC since ADAM will not accept a number as a variable name, and it would not be able to distinguish between a line number and a value being assigned to a variable. Similarly, the arrowhead under the plus sign is ADAM's reminder to you that a plus sign can only follow an equal sign under the grammatical rules of SmartBASIC.

Choosing Names for Variables

In choosing names for your variables you are limited by a few simple rules.

1. Variable names can start only with a letter. For example: C, C5 and CH are valid names. 3H is not.
2. Variable names can contain only numbers and letters. They cannot contain any other symbol or character, except for the dollar sign (which can end the variable name and which creates a *string* variable to be discussed soon). A%, D* and 7& are not acceptable names.
3. Only the first two characters (and the last if it ends with a dollar sign) of the variable's name are read by ADAM. For example, type:

```
]mountain = 5000
]molehill = 50
]print mountain
```

ADAM will respond with the number 50 since it sees only the "mo" in each of the variables. ADAM believes that you first gave MO a value of 5000 and then changed your mind and decided to make MO equal to 50. So when you asked ADAM to print the value of MO, it printed the last value you gave—50. ADAM cannot tell the difference between a mountain and a molehill. This is why the number of variable names that you can use is so limited. You can calculate that ADAM can distinguish only 2592 variable names, exactly half of which are for the mysterious string variables and their dollar signs.

4. Choose variable names that remind you of the things they represent. ADAM won't mind if you use long variable names. If you use many variables in a program, it's much easier to keep track of them if their names hold some meaning for you. For example, use names like VELOCITY, ELEVATION or PROFIT to represent the speed of a bullet, the height of a building, or the amount of money you made in your last stock transaction. These are much easier to remember than VE, EL, and PR, or X, Y and Z. Using easily recognizable names makes programming easier, especially when you go back to finish or read a program a few days later. It's a habit you should develop.

Programming Considerations

Variables are very important for programming. The most immediate implication is that with numeric variables it's no longer necessary to repeat the same number many times in a program, even if that number will be used many times. Instead, one can store the number in memory by using a variable. Then, whenever you want to work with the number, use the variable name instead. It's easier and briefer that way. Watch:

Suppose you were given $50,000 to buy a Mercedes Benz costing $14,344. Of course, if you were willing to part with all your money, you could purchase more than one Mercedes. To see how many of these cars you could buy, consider the following program:

```
]10 cost = 14344
]20 print cost
]30 print cost * 2
]40 print cost * 3
]50 print cost * 4
]60 end
```

The effect of line 10 of this program is to store the price of a single car in a specific memory location inside ADAM, and to label that location COST. COST is, in effect, a variable identifying a particular memory location. Line 20 has ADAM print the price of one car. In lines 30, 40, and 50, ADAM computes the price of two, three and four cars, respectively, and prints the results out on the screen. If you were to write this program and type RUN, you would get the following:

```
14344
28688
43032
57376
```

As you can see from the results, your $50,000 endowment would allow you to buy only three Mercedes Benz autos. How cruel life can be! The larger point, however, is to recognize that it is possible, using a variable, to store a specific number in memory and then simply refer to that variable name when you want to use the number. This can be a big time and space saver, as the following program illustrates.

```
]10 dozen = 12

]20 print "In 1 dozen donuts th
ere are " dozen " donuts."

]30 print "In 2 dozen donuts th
ere are " dozen * 2 " donuts."

]40 print "In 3 dozen donuts th
ere are " dozen * 3 " donuts."

]50 print "In 4 dozen donuts th
ere are " dozen * 4 " donuts."

]60 print "In 5 dozen donuts th
ere are " dozen * 5 " donuts."
```

The program above will compute just how many donuts you have for batches ranging from one to five dozen. The nice feature about the program is that you have to key in the critical value for a dozen only once. The computer has stashed away the number 12 in one of its numerous memory locations, and anytime you wish to refer to it you simply use the variable name DOZEN. ADAM then immediately knows you're talking about the number 12. You could then program ADAM to acknowledge the fact that some dozens contain 13 donuts—baker's dozens! How could you do this? It's simple. All you would need to do is rewrite line 10 so that it read:

```
]10 dozen = 13
```

By retyping line 10, you are in effect changing the value in the memory location identified by the variable DOZEN. Instead of having the number 12 stored in this electronic pigeonhole, ADAM now has the

number 13. To see that this is so, try reviewing your program by typing the LIST command followed by a carriage return. Line 10 should now say "dozen = 13." Even more importantly, when you run the program the new results should reflect the fact that dozens are now baker's dozens. It is easy to change the value represented by a variable; simply define the variable again, giving ADAM the new value.

String Variables

Now that you've expanded your computer vocabulary to include numeric variables, you probably suspect that there are variables for letters and words as well—variables representing locations where things other than numbers are stored. If you're thinking that, your suspicions are right. SmartBASIC has string variables that do precisely that.

In computer circles, a string is simply a word. More precisely, a string is anything that ADAM hasn't been programmed to think of as a number. This includes letters, words, sentences, spaces and punctuation marks, all of which must be enclosed in quotation marks to be considered strings. Strings can even consist of numbers—provided, of course, that they too are enclosed in quotation marks.

It follows logically, then, that a string variable is a variable representing a location in ADAM's memory where strings are stored. However, we can't simply use a letter to name a string variable. The letters, you'll recall, have already been reserved for numeric variables. Instead, string variables are named using letters and words with a $ tacked onto the end. You can think of the dollar sign as a fancy S standing for string variables. A few examples should illustrate this.

Let's put the words from an Irving Berlin song into a string variable named C$.

```
]10 C$ = "I'm dreaming of a whi
te Christmas, just like the ones
I used to know."

]12 print C$

]20 end
```

If you tell ADAM to RUN, the first line of "White Christmas" will be printed on your screen. For example:

```
]run

]I'm dreaming of a white Chris
tmas, just like the ones I used
to know.
```

Now suppose we'd like to add a line from the song "Let It Snow" to our program. We might type the following:

```
]15 Let E$ = "Oh, the weather outside is frightening."

]17 print E$
```

If you list your program, you'll find that ADAM has dutifully inserted lines 15 and 17. What's more, if you run your program a second time, you will find both Christmas songs being printed on your screen, exactly as you wished.

What's happening is that ADAM has taken the line from "White Christmas" and placed it in a memory location labeled C$. When you type PRINT C$, ADAM immediately prints whatever is currently found in memory location C$—in this case, the Irving Berlin song.

The same can be said of E$. Like C$, E$ is the name for a place in ADAM's memory where words are stored—in this case, our second Christmas song. When you tell ADAM to print E$, it scrounges around in its memory for the location marked E$ and prints whatever lies there. As long as there is something inside location E$ (and in this case we have assigned words to the location), ADAM will dutifully return the contents of E$.

This raises another point. Ordinarily, ADAM will not give you an error message if you ask it to print the contents of a variable that hasn't been defined. Instead, ADAM will print zero for undefined numeric variables and a blank space for undefined string variables. Type NEW, wait for the bracket to reappear on your screen, and then type the following program:

```
]10 print A
]20 print B$
]30 end
```

Now type RUN.

Since neither A nor B$ has been defined, the contents of the memory locations labeled A and B$ are empty. Your output should be the following:

```
0
```

ADAM returns a zero to reflect the contents of the numeric variable and a blank line indicating that the string variable has not yet been defined. By now, however, you should recognize that with a single command it's possible to create substance where formerly there was a void. You might add the following to your program:

```
]5  A = 7
]7  B$ = "Merry Christmas."
```

Now try running your program. You should get:

```
7
Merry Christmas.
```

Suppose you feel sorry for ADAM, because computers cannot share the joy and emotions of festive occasions. Let's see if we can teach ADAM some of the fundamentals of being human. Start by typing:

```
]love$ = "chocolate, a Mercedes
Benz, good friends, and a new
8-bedroom house."
```

Now see what happens if someone asks ADAM to describe its definition of love:

```
]print "Love is "love$

Love is chocolate, a Mercedes B
enz, good friends, and a new 8-
bedroom house."
```

Some might think that ADAM's new owner is a bit too materialistic.

Notice that when you assign a value to a string variable, you must enclose the string of characters in quotation marks. This allows you to include spaces at the beginning and end of the characters you wish to include in your string variable. For example:

```
]hate$ = "trips to the dentist
, the tax man, . . . . and my old
tennis shoes. "
```

The spaces outside the quotation marks are ignored, as are the letters te in hate$. The spaces within them, however, are part of the string variable. Just as with numeric variables, ADAM can't read beyond the first two letters. It can't tell the difference between HATE$ and HAIRY$ or between

POWER$ and POLECAT$. ADAM does recognize them as string variables, however, and will not confuse them with numeric variables. Try typing:

```
]power = 20
]polecat$ = "Bill Maloy"
]print power$
```

ADAM responds with:

```
Bill Maloy
```

When you told ADAM to print POWER$, ADAM searched for a string variable beginning with the letters PO. Because of the dollar sign, ADAM did not look for a numeric variable beginning with those same letters. Computers consider numeric and string variables to be very different entities. You can't even use them in the same equation. Typing

```
]prince$ = polecat$
```

is perfectly acceptable (unless you know Bill), but typing

```
]skunk = polecat$
```

will result in:

```
skunk = polecat$
              ^
Numeric Equation Expected
```

It matters little whether a skunk is a polecat or not. ADAM won't accept that conclusion because computers just get stubborn about this sort of thing. They just won't let you mix apples and oranges (metaphorically speaking), or string variables and numeric variables (literally speaking).

Now for something that will start your creative juices flowing. You can add string variables. (No, you're not being strung along.) You can't divide, subtract or multiply strings, but you can add them. What do you get? Try this:

```
]love$ = "I love you because "
]hate$ = "you broke my heart."
]marriage$ = love$ + hate$
]print marriage$
```

You'll get

```
I love you because you broke my
heart.
```

You could also type:

```
]marriage$ = love$ + "you broke
 my heart. "
]print marriage$
```

and get the same result. You add string variables to any text or group of characters as long as you put quotation marks around the text. However, you can't have any numeric variables or values in an equation in which you are adding string variables. Typing:

```
]Romeo = 17
]Juliet$ = "a lovely young girl of"
]love$ = Juliet$ + Romeo
```

results in an error.

```
love$ = Juliet$ + Romeo
                  ^
Type mismatch
```

ADAM is not trying to say that Romeo and Juliet aren't right for each other (though hindsight indicates that this might have been good advice). ADAM, stubborn little machine that it is, just refuses to mix string and numeric variable types, no matter how hard you try or how often you command.

At this point you've probably decided that ADAM is not much of a romantic. If you've given up on filling up ADAM's head and you want to erase all the variables that you defined this session, simply type the command CLEAR and ADAM will forget every variable—numeric and string—that you have told it to remember. CLEAR will not delete programs that you have created; the NEW command you read about in the last chapter is reserved for that purpose. Likewise, NEW will not affect ADAM's memory of the values of variables. Neither will the HOME command, which erases only the screen.

String Functions

Suppose you still think that Romeo and Juliet are right for each other, even though ADAM insists that they are not. (ADAM is beginning to sound like some parents.) If you, as Juliet, really want to be with Romeo, you'll just have to convince ADAM (or your parents) that Romeo has reformed, that he has given up his old ways. To do this, type:

```
]Romeo = 17
]Juliet$ = "a lovely young girl of"
]love$ = Juliet$ + str$(Romeo)
]print love$
```

and you get:

```
a lovely young girl of 17
```

Typing STR$( ) looks as if you're putting Romeo in a straight-jacket. (After a few years, he might begin to think so.) Actually, you just told ADAM that the love of Romeo and Juliet wouldn't result in a "mixed marriage." The STR$ command converts numeric variables or even plain numbers into strings. These strings are perfectly compatible with string variables or other strings of text that you've enclosed in quotation marks.

Suppose you think it's unfair that Romeo should be the one to change. Maybe Juliet is at fault, since she does have a bad habit of inviting young men onto her balcony. She did offer to change if Romeo would not. (Look it up.) If you inform ADAM of this change by typing,

```
]Juliet = val(Juliet$)
]love = Juliet + Romeo
]print love
```

you'll get the number 17. The VAL command is nearly the complement of the STR$ command. It asks ADAM for the numerical value of the string in the parentheses. If the first character of the string is not a digit, plus sign or minus sign, then ADAM gives it a value of zero. If the string begins with a positive or negative number, ADAM assigns it the value of that number—the number that you will get by starting at the beginning of the string and continuing until you reach a non-digit character. For instance, VAL("−123 456") is −123, VAL("345+67") is 345, and VAL("A bird in the hand.") is zero.

In the example of our star-crossed lovers, the numeric variable JULIET is assigned the value zero, since the string variable JULIET$

doesn't begin with a digit, plus sign or minus sign. Therefore, the numeric variable LOVE becomes the sum of zero and the value of the numeric variable ROMEO—17.

You are appalled by ADAM's conclusion that JULIET contributes nothing to the value of LOVE, that ROMEO is contributing everything. That will teach you to ask a computer how to measure love. Seventeen is the sort of answer one expects from a computer. Let's see if ADAM can measure love's value in another way. Try this example:

```
]Juliet = len(Juliet$)
]love = Juliet + Romeo
]print love
```

This time, you get an answer of 40. It seems that their love is growing rapidly. Since you already know that ROMEO contributes 17, JULIET must be making up the difference. In fact, JULIET contributes 23 now. If you look carefully, you will see that there are exactly 23 characters including blanks in the string variable JULIET$. The LEN command comes from an abbreviation of the word "length." It simply counts the number of characters in a string. You can even count the number of digits in a numeric variable if you use the STR$ command. This trick works only for nonnegative integers, since decimals and minus signs would also be counted. For instance, LEN(STR$(4567)) is 4, but LEN(STR$(−45.67)) is 6.

The remaining two string commands are quite similar. The first is the LEFT$ command; the second is the MID$ command. Both allow you to extract characters from strings or string variables. As you will soon see, the LEFT$ command is only a special form of the MID$ command.

To see how the MID$ command works, begin by typing:

```
]family$ = "Dad, Mom, Junior"
```

Now that ADAM knows your family, you want your new computer to say hello to your mother. Somehow you must instruct ADAM to pull your mother's name out of the string. You can do this with the MID$ command. Just type:

```
]mother$ = mid$(family$, 6, 3)
]print "Hello "mother$
```

ADAM will then say:

```
Hello Mom
```

Aren't you impressed? Your protégé is coming along quite nicely. If only you could teach him not to hog the television set.

When using MID$, you need to supply three parameters—the string or name of the string variable, a place to begin, and the number of characters. In this example, a string variable was named—FAMILY$. Then ADAM counted over six characters from the beginning of the string to the "M" in "Mom." ADAM then counted over three characters to the end of "Mom." The PRINT command told ADAM to display those three characters on the screen.

Try this again, greeting your father instead.

```
]father$ = mid$(family$, 1, 3)
]print "Hello "father$
```

To this your trusty computer replies:

```
Hello Dad
```

ADAM will respond in exactly the same fashion if you type:

```
]father$ = left$(family$, 3)
]print "Hello "father$
```

The LEFT$ command tells ADAM to extract the leftmost characters from a string. In the parentheses you must define the string and the number of characters. As you've guessed, the LEFT$ command is merely a MID$ command beginning at the first character (the middle parameter = 1), from which point ADAM will start reading.

Just one more example and you'll know all you need to know about string functions. Type:

```
]myself$ = mid$(family$, 11)
```

Notice that you are deleting the third parameter. Now type:

```
]print "Hello "myself$
```

ADAM will finally greet you with:

```
Hello Junior
```

There is really just one more thing to know about this command. The third parameter is optional. If you leave it off, ADAM simply extracts all of the string following the beginning position designated by the second parameter.

Arrays

For very large programs, you may soon grow tired of thinking up new and original variable names. There is a solution to this problem, called an array. An array is a systematic grouping of items or an arrangement of elements into rows and columns. That meaning is valid when used with reference to computers, too.

For instance, suppose you want to save a list of the names of the people with whom you work or go to school. You can use the variable NAME$ to refer to the entire list whenever you communicate with ADAM. Or suppose that you are a school principal and need to keep track of the names of the teachers in each of your classrooms.

So that ADAM won't be caught off guard when you want to use an array, you must type the DIM command (short for "dimension") to instruct ADAM to set aside a portion of its memory for your array. This command should appear very near the beginning of your program. To illustrate this, let's pretend that you're the school principal, responsible for 25 classrooms. Begin by typing:

```
]10 dim room$(25)
]20 room$(1) = "Mr. Brown"
]30 room$(2) = "Ms. Faucette"
]40 room$(3) = "Ms. Kingsbery"
]50 room$(4) = "Mr. Hammer"
]60 room$(5) = "unassigned"
```

Anytime in the future, you can find out which teacher has been assigned to which room by typing RUN and then PRINT ROOM$(X) where X is the room number.

Arrays can be composed of either numbers or strings, just like numeric variables and string variables. The rules are the same. You just have to tell ADAM in advance how many elements will be in the array. It's as if you are making reservations for a hotel room. Just tell ADAM to reserve a place in its memory big enough to accommodate the information you plan to type in later. If you don't know the exact number, just "overbook." It's better to reserve a space too large than one too small.

The schoolteacher example above used a one-dimensional array—named ROOM$—which was composed of 25 elements. The number within the parentheses is called the index. You can refer to specific items of information (the names of the teachers) by the name of the array and the index.

Due to your success at running the high school, the school board has decided to increase the size of your school by adding three floors to your old one-floor building. These floors also have 25 rooms each, giving you a total of 100 rooms for which you are responsible. You can use your old program by typing:

```
]10  dim room$(100)
```

or you can create a two-dimensional array with two indexes—one referring to the floor number, the other to a particular room on that floor. You do this by typing:

```
]new
]10  dim room$(4,25)
```

and then

```
]20  room$(1,1) = "Mr. Brown"
]30  room$(1,2) = "Ms. Faucette"
]40  room$(1,3) = "Ms. Kingsbery"
]50  room$(1,4) = "Mr. Hammer"
]60  room$(1,5) = "unassigned"
```

to reassign the teachers to their old rooms, and

```
]70  room$(2,21) = "Mr. Ward"
]80  room$(3,3) = "Ms. Landers"
```

to make assignments for two of your new staff members.

You could even add a third index, creating a three-dimensional array, to account for the school quarter for which that teacher is assigned to that room.

```
]dim room$(4,25,4)
```

Typing:

```
]80  room$(2,21,3) = "Mr. Ward"
```

would indicate that Mr. Ward had been assigned room 21 on the second floor for the third quarter.

ADAM can handle one, two, and three-dimensional arrays as long as the total number of elements in the array doesn't exceed the capacity of its memory. Limit the size of your arrays to about four thousand elements, depending on the dimension of the array and whatever else you've told

ADAM to remember. You can calculate the number of items or elements in the array by multiplying the indexes in the dimension statement together. For example, DIM ROOM$(4,25) creates 100 locations and DIM ROOM$(4,25,4) tells ADAM to set aside 400 locations.

You now know how to write simple programs that will do numerical calculations and keep vital data in ADAM's pigeonholes, which you've labeled with your variable names. If you don't feel comfortable with these new skills and concepts, don't worry. You're learning a new language and aren't expected to understand everything all at once. Relax and continue reading. Then go back and review the concepts again. After a while you'll feel quite at ease with SmartBASIC.

CHAPTER 7
INPUT AND OUTPUT

A classically educated person would probably think that I/O (pronounced eye-oh) referred to yet another maiden pursued by Zeus. Unfortunately, that person would be wrong; Io's affair with Zeus had little to do with computers. I/O doesn't refer to one of Jupiter's moons either. To the computer literate, I/O stands for Input/Output—everything pertaining to communicating with your computer, shoveling data and commands into its metallic mind and shoveling information and calculations back to the world outside.

You've already dealt with computer I/O if you've ever used a computer, a video game or a telephone. Typing on a keyboard, moving a joystick and dialing a telephone number are all methods of inputing information into a computer. The screen displays generated by your computer or game are examples of the computer transmitting information back to you. However, ADAM doesn't always have to send data to you directly or by way of the screen. Your computer can send information to external storage areas, so you can keep it for future use, or to the printer, so you can have a *hard copy* (a printed copy of the information).

The Digital Data Pack

When using SmartBASIC, you must type the SAVE command to instruct ADAM to send information to its digital data pack. If you remember, you would press the STORE/GET key if you were using SmartWriter. Unfortunately, pressing that key has no effect when using SmartBASIC. After you type SAVE, you must also type a space and a filename before hitting the return key. A copy of the program on which you are currently working will be stored on the data pack as a file, and will be referenced by the filename that you designated. You can use any characters, but no more than ten. For example, type the following program into ADAM:

```
]new

]10 print "To be, or not to be:"

]20 print "That is the question
."

]30 print "Whether tis nobler i
n the mind"

]40 print "to suffer the slings
 and arrows of outrageous fortu
 ne."

]50 end
```

Now store this program by typing:

```
]save Hamlet
```

As soon as you press RETURN, the tape in the data pack drive will begin to spin and ADAM will transfer your program onto the data pack. Now you have to be patient. This will take a while. If you are storing a large program, you might want to run to the kitchen and grab something to eat. Computer memory based on tape is inexpensive but slow.

Type NEW to clear ADAM's memory, then type RUN just to make sure. You should get no response from ADAM. ADAM has forgotten the Danish prince's soliloquy. To refresh ADAM's memory, you must type LOAD, followed by the filename HAMLET. The LOAD command tells ADAM to retrieve the designated file from the external storage area (the digital data pack), and place a copy of it into its internal memory. If you type RUN now, you should get:

```
To be, or not to be:
That is the question.
Whether tis nobler in the mind
to suffer the slings and arrow
s of outrageous fortune.
```

Sir Laurence Olivier had better beware.

Suppose that you want to increase ADAM's Shakespearean repertoire. Begin by typing:

```
]new
]10  print "Tomorrow, and tomo
rrow,"

]20  print "and tomorrow, Creeps
in this"

]30  print "petty pace from da
y to day"

]40  print "To the last syllabl
e of"

]50  print "recorded time,"

]save Macbeth

]new

]10  print "If I profane with m
y"

]20  print "unworthiest hand th
is holy"

]30  print "shrine, the gentle
sin is this"

]save Juliet
```

Now ADAM knows parts of three great plays. Before you realize it, you may find ADAM asking your permission to join an acting company. Before ADAM does any serious performing, you might want to let it in on a little secret—the last speech belonged to Romeo, not Juliet. It sure is easy to fool a computer; it may not be nice, but it is certainly easy. Let's correct poor ADAM's memory.

You can see the mistake by typing CATALOG. After a minute or so, ADAM will display a list of all of the programs that you've saved on the data pack in the drive. In this instance you should see the filenames Hamlet, Macbeth and Juliet listed along with any others you might have created. Now it's time to replace the erroneous filename. Type:

```
]rename Juliet,Romeo
]catalog
```

The RENAME command changes the old filename, Juliet, to the new one, Romeo. It does not affect the contents of the file in any way. When

ADAM prints the new file catalog, you should no longer see Juliet listed. You should see only Hamlet, Macbeth and Romeo.

It's a few weeks later. The critics' opinions are sitting on your desk. ADAM's portrayal of Hamlet and Macbeth went well, but in the role of Romeo, it was less than convincing. I suppose that ADAM is not a romantic figure. Maybe the definition of love that you gave ADAM in the last chapter has something to do with it. It's best for unsuspecting audiences that ADAM forget the part of Romeo. You can do this by telling ADAM to erase the file named Romeo. Do this with the DELETE command:

```
]delete Romeo
```

Now type:

```
]catalog
```

You will notice that ADAM now has no recollection of the file Romeo. Do not confuse DELETE with DEL, which will be mentioned later. The former must be followed by a filename, the latter by a line number.

You can delete all of the information on a digital data pack, including SmartBASIC, by typing INIT HELLO. Don't ever try this with the Smart-BASIC tape in the drive. Use it only on a data pack that you want to wipe completely clean.

Suppose you want to expand ADAM's role as Hamlet. Type:

```
]new

]10 print "Alas, poor Yorick!"

]20 print "I knew him, Horatio
, a fellow"

]30 print "of infinite jest of
most excellent fancy."

]save Hamlet

]catalog
```

This new program will not be added to the old Hamlet file; it will replace it. Whenever you type SAVE and the filename of an existing file, the program in ADAM's internal memory is written over the old one, but the old one is also saved. Notice the letters to the left of the filenames. You should see a column composed of the letters A and a. The small a designates the old file; the capital A designates the one that you just

saved. If you save HAMLET once more, the file designated with the capital A is written over the file with the small a, and the file that you just saved is written over the file that had the capital A. The oldest version of HAMLET, the one that had been next to the small a, would now be lost. You could never load it again.

If you decide that you don't want to lose the oldest version of HAMLET, you need to change the small a into a capital A. Use the RECOVER command to perform this task. RECOVER will have no effect if there is currently a file named HAMLET with a capital A next to it. You must first RENAME or DELETE the file next to the capital A and then use RECOVER.

To prevent accidents from occurring, you can use the LOCK command. You cannot delete a file that's been locked, nor can you save a file with that filename. If you attempt either, ADAM will chastise you with an error message. You can preserve the "To be, or not to be:" soliloquy with:

```
]lock Hamlet
```

Then if you try to save the new program, ADAM will respond with:

```
FILE LOCKED
```

You would then have to save "Alas, poor Yorick!" under a different filename. If you type CATALOG at this point, you will notice that an asterisk has suddenly appeared next to the A near the filename HAMLET. This asterisk indicates that the file has been locked.

To unlock a file, simply type UNLOCK and the filename. The file will be exactly as it was before you locked it. Typing:

```
]unlock Hamlet
```

will allow you to update your file.

If you get tired of typing:

```
]new
]load Hamlet
]run
```

you can simply type:

```
]run Hamlet
```

The effect is exactly the same. If you just type RUN, the program currently in ADAM's internal memory will be activated. RUN followed by a filename erases the current program, loads the new one and starts it going.

Interactive Programs

With the proper instructions ADAM can stop during the execution of a program and ask for information from you before continuing. ADAM's solution or output can vary, depending on the information you provide. Such programs are called *interactive* programs. They are very useful in solving many difficult and complex problems that require the judgment and knowledge of the user at critical decision points.

There are two SmartBASIC statements that cause the program to stop until a question has been answered. The first of these is the GET statement. The second, and more useful, is the INPUT statement.

The GET statement allows the operator (the person sitting at the keyboard), to input a single character from the keyboard at a particular point in the program. The character that is typed will not appear on the screen, and pressing the return key is not necessary. ADAM will consider the character either a string value or a numeric value, depending on the type of variable preceding the GET statement in the program. The GET statement is most often used when the question asked of the operator is a multiple choice question, or one requiring a simple yes or no answer. It is used almost exclusively in conjunction with the IF statement, which you will learn about in the following chapter. Examples of its use can be found in that chapter also.

The INPUT statement tells ADAM to print a question mark and to wait until you hit the return key to continue. The information that you provide can be one or more numbers or strings. Try this example:

```
]new
]10  print "What is your name?"
]20  input name$
]30  print "How old are you?"
]40  input myself
]50  print "How old is your fat
her?"
]60  input father
]70  print "How old is your mot
her?"
]80  input mother
```

```
]90  print "You're a fine fello
w, "name$"."

]100  print

]110  print "When you were born
, your father was "father-mysel
f" years old,"

]120  print "and your mother wa
s "mother-myself" years old."

]run
```

ADAM will ask you a series of questions for you to answer. (Sample responses are given.)

```
What is your name?
?Alex
How old are you?
?12
How old is your father?
?36
How old is your mother?
?34

You're a fine fellow, Alex.
When you were born, your fathe
r was 24 years old
and your mother was 22 years o
ld.
```

With a pair of statements—DATA and READ—you can store information in the program for use when you run it. The information is stored before or after the main body of the program in a series of lines beginning with the word DATA. You can use any number of them and each can contain either numbers or text. With the READ statement, ADAM will assign the information stored in the DATA statements to variables. By putting commas between the pieces of information, one DATA statement can provide values for many variables. For instance:

```
]new
]10 data 10, 20, 30, 40
]20 read a, b, c, d
]30 print "A IS ";a
]40 print "B IS ";b
]50 print "C IS ";c
]60 print "D IS ";d
]end
]run
```

ADAM will print:

```
A IS 10
B IS 20
C IS 30
D IS 40
```

ADAM will read the information in the DATA statements one piece at a time, from left to right. If you tell ADAM to read more times than there are pieces of information between the commas on all of the DATA statements, ADAM will stop the program and give you an error message telling you that it has run out of data.

You can use the same data over and over, however, with the RESTORE statement. This tells ADAM to start again with the first DATA line when READ tells it to look for more information. Let's alter the last example to take advantage of this. Type in these new lines to add to the program:

```
]10 data 10,20
]15 read a,b
]20 restore
]25 read c,d
]run
```

This time ADAM will print:

```
A IS 10
B IS 20
C IS 10
D IS 20
```

The most common use of DATA and READ is assigning large amounts of information to the arrays that you create with the DIM command. To do this efficiently, you will need to use a FOR . . . NEXT loop. You will learn what this is and how to apply it in the next chapter.

Turning the Printer On

Not only can ADAM communicate to you through the screen, it can also print onto paper, generating a hard copy. Just type PR#1 to activate the printer. Now everything that ADAM prints on the television screen will also be printed on the printer. One side effect of using the printer is that ADAM will slow down, since the printer is slower than the screen. When you want to turn the printer off, just type PR#0.

Print Formatting

You can improve the appearance of a program's output with a few simple tools. The process of arranging this output so that it is easier to read and understand is called formatting. Several characters, functions and statements make this possible. They include the semicolon, the comma, the SPC, TAB and POS functions, and the INVERSE, NORMAL and SPEED statements.

To get started, you might want to learn a trick that will save you some time. Rather than type the command PRINT over and over again, ADAM will let you type a question mark instead. ADAM interprets ? exactly as it interprets PRINT. In fact, if you list the program after you type it in, ADAM will replace your ?s with PRINTs. As an added convenience, notice how close on the keyboard the question mark is to the quotation mark and shift key, both of which are used very often. Feel free to use this shortcut whenever applicable.

A semicolon placed at the end of a print statement tells ADAM to begin printing at the next character position. This effectively joins together any text to be printed. For example, typing:

```
]new
]10 ?"con"
]20 ?"cat"
]30 ?"e"
]40 ?"nation"
]run
```

results in:

```
con
cat
e
nation
```

However, typing:

```
]new
]10 ?"con";
]20 ?"cat";
]30 ?"e";
]40 ?"nation"
]run
```

will give you:

```
concatenation
```

As you can see, the semicolon concatenates data by displaying strings or numbers next to each other, with no spaces between the values.

The comma, on the other hand, separates printed data into fields. In SmartBASIC, a comma will place output into two fields, one on the left side of the screen, and one on the right. Typing:

```
]new
]10 ?"con",
]20 ?"cat",
]30 ?"e",
]40 ?"nation"
]run
```

will result in:

```
con             cat
 e              nation
```

Typing:

```
]new
]10 ? 55,555,5555,55555,555555,
5555555,55555555,555555555
]run
```

results in:

```
55              555
 5555           55555
 555555         5555555
 55555555       555555555
```

Notice that the first item printed in each example is offset. The left boundary of the leftmost field is one space to the right of the first column. You can correct this aesthetic defect by printing two blanks before printing your values. For example:

```
]new
]5  ? " ","  ",
]10  ? 55,555,5555,55555,555555,
5555555,55555555,555555555
]run
```

gives:

```
55              555
5555            55555
555555          5555555
55555555        555555555
```

The semicolon and comma give you some control over the format of your output, but for more elaborate displays, you can use one of the functions that is designed to work with the PRINT command. The first of these is the SPC function.

SPC tells ADAM to "space over" a given number of spaces. It is included in the PRINT statement following the PRINT command, and must be followed by a numeric value in parentheses. That value tells ADAM how many spaces to skip before printing, and shouldn't exceed 30, since ADAM displays only 31 characters per line in SmartBASIC.

The next function that you might want to use is TAB. This function works much like the tabbing of a typewriter, or the tabbing in SmartWriter. TAB is often more useful than SPC because it tells ADAM the exact horizontal position at which to begin printing. You don't have to count spaces or positions in order to create even columns of information. As with SPC, you need to enclose a number from 1 to 30 in parentheses after TAB. That number will be the position at which ADAM will print. Here is a program that illustrates the differences between SPC and TAB.

```
]new
]10 ? "ACCOUNTS OUTSTANDING"
]20 ? "-----------------------
-------"
]30 ?"H. ROARK";spc(11)
]35 ?"$123.45"
]40 ?"D. TAGGART";spc(9)
]45 ?"$222.78"
]50 ?"J. GALT";spc(12)
]55 ?$314.99"
]60 end
```

When you run this program, you will get:

```
ACCOUNTS OUTSTANDING----------

H. ROARK          $123.45
D. TAGGART        $222.78
J. GALT           $314.99
```

The only problem is that you have to recalculate the number to put after SPC in each line in order to make the right column line up. You will always have to do this when the items in the first column aren't all the same length. Using TAB is much easier. For instance:

```
]10 ? "ACCOUNTS OUTSTANDING"
]20 ? "-----------------------
-------"
]30 ?"H. ROARK";tab(20)
]35 ?"$123.45"
]40 ?"D. TAGGART";tab(20)
]45 ?"$222.78"
]50 ?"J. GALT";tab(20)
]55 ?"$314.99"
]60 end
```

Again, type RUN and you get:

```
ACCOUNTS OUTSTANDING
-------------------

H. ROARK            $123.45
D. TAGGART          $222.78
J. GALT             $314.99
```

This time you didn't have to do any counting yourself. ADAM moved to the twentieth position automatically.

Another PRINT function will tell you the location of the cursor. It's called POS (for "position") and it tells ADAM to determine the number of the column at which the cursor currently resides. It is also followed by a number in parentheses, but this time the number has no meaning. It's called a dummy value. A zero is usually used. For example, type:

```
]new

]10  ?"THE CURSOR IS IN COLUMN "
;pos(0)

]20  end
]run
```

and ADAM will print:

```
THE CURSOR IS IN COLUMN 24
```

So much for POS. You probably won't have need for it very often.

If you really want to improve the looks of your output, consider the INVERSE command. When ADAM sees this, everything that is subsequently printed on the screen appears like a photographic negative of the normal text. Instead of the white characters on a black screen that you see when you first turn ADAM on, the printing will be black letters on a white background. You can switch back to the standard mode by typing NORMAL. The inverse characters will remain on the screen until you type HOME or until they are scrolled past the top edge. You can use both of these commands within your programs to switch back and forth. Try this for practice:

```
]new
]10  ?"I AM NORMAL."
]20  inverse
]30  ?"I AM INVERSE."
]40  normal
]50  ?"I AM NORMAL AGAIN."
]60  end
]run
```

You can also control how fast ADAM prints characters on the screen with the SPEED statement. Your range of options includes any whole number between 0 (slowest) and 255 (fastest). Try this program:

```
]new
]10 speed = 10
]20 ? "HERE I AM GOING UP THE H
ILL."
]30 speed = 30
]40 ? "I'M ON THE WORLD'S FIRST"
]50 speed = 70
]60 ? "COMPUTER ROLLER COASTER."
]70 speed = 120
]80 ? "I'M AT THE TOP, AND HERE
I GO."
]90 speed = 180
]100 ? "BOY, THIS IS FUN."
]110 speed = 255
]120 ? "BUT, I'M GLAD"
]130 speed = 100
]140 ? "IT'S FINALLY"
]150 speed = 20
]160 ? "OVER."
]170 end
```

Editing and Debugging

When you're working in SmartBASIC, you don't always have to retype a program line whenever you want to make changes to it. All you have to do is put the cursor at the beginning of the program line and make the appropriate changes before you press RETURN. ADAM will type over all of the old characters as you type in new ones. You can use the left and right cursor arrows to move along the line without destroying it. Be sure that you don't use the other cursor-control arrows and that you don't move off the program line when editing it. Now move the cursor just past the end of the line and press <RETURN>. If you press <RETURN> twice when the cursor is in the middle of the line, everything after the cursor will be deleted.

If you can't see the program line that you want to edit on the screen, tell ADAM to print it out using the LIST command. If you just type LIST, ADAM will print the entire program. If you put a line number after LIST, ADAM will print only that line. You can also tell ADAM to print several lines at once by typing LIST XX – YY, where XX is the first line number and YY is the last. ADAM will print both of these lines and every line in between. You can also use spaces for XX and YY. This means list "from the beginning" or "to the end."

If you want to delete a line, type the line number and then press <RETURN>. Another method that allows you to delete one or more program lines is to type DEL followed by a line number or a range of line numbers, just as you did with the LIST command.

You will often need to delete or add spaces when you're editing a program line. To do this you need to learn how to use ADAM's <CONTROL> key. When you hold down <CONTROL> and press another key, no characters are printed, but ADAM is not ignoring you. If you hold down <CONTROL> and press N when editing a line, a space will appear. This command is called <CONTROL>N. Now try <CONTROL>O several times. This command will cause ADAM to delete characters from the program. If the program is longer than one line across the screen, notice that <CONTROL>N and <CONTROL>O will not affect the characters on the second line. If you press <CONTROL>N many times, the characters to the right of the cursor will be pushed off of the right edge of the screen. These characters will not return, even if you delete the spaces that you just added. Keep this in mind when editing.

There are several other important <CONTROL> functions. <CONTROL>L has the same effect as typing HOME; it will clear the screen. <CONTROL>P tells ADAM to print a hard copy of everything that appears on the screen. Unlike PR#1, <CONTROL>P doesn't stay on until you turn it off. It simply prints until the printed page looks like the

screen, then stops unless you press it again. When text is scrolling by too fast for you to read, as it might when you type LIST, <CONTROL>S tells ADAM to stop printing. Hit <CONTROL>S once more and ADAM will start printing again. <CONTROL>C tells ADAM to stop running a program. If ADAM is waiting for you to input data, then just type <CONTROL>C followed by <RETURN>. ADAM will stop executing the program and print "Break In Line XX" where XX is the line where ADAM stopped the execution of the program.

You may often need to follow ADAM's path through a program in order to find a problem with it. The best way to do this is to type TRACE before you run the program. ADAM will then print a list of the program line numbers as each line is executed. If the program is long, you might want to turn on the printer first, since the information will quickly scroll past the top edge of the screen. The TRACE command will continue to be activated until you type NOTRACE. You can then run your program as you normally would.

SmartBASIC and SmartWriter Together

There is one more thing you need to know. SmartWriter and SmartBASIC use the same files. This lets you edit SmartBASIC programs in SmartWriter if you wish. Just remove the data pack and reset ADAM. Then use the same procedure, beginning with the STORE/GET command key, to load the file into ADAM's work space. When you finish, store the program as you would store a normal text file. The main disadvantage of this method is that ADAM won't point out your errors, as in SmartBASIC. If you are relatively sure that you know the proper formats of the commands, however, this method might be for you. The major advantages are better control over editing and the ability to see more of the program on the screen.

In addition, you can print your program on the printer while you're using SmartWriter. You can even change the margins, add other text, or incorporate the program into a report or letter. Most important, you can merge SmartBASIC programs by using STORE/GET to move them into the work space, edit them to remove duplicate line numbers, and then store the whole thing under one filename.

SmartBASIC also allows you to perform commands on SmartWriter files. Although there is no RECOVER command in SmartWriter, you can recover files created with SmartWriter when SmartBASIC is running by

using both the DELETE and RECOVER commands, just as you would with a program you had saved in SmartBASIC. When a digital data pack gets close to full, it's a good idea either to get a new one or to erase old files before you find yourself with a file in working memory or which ADAM doesn't have enough room to save.

You have now learned everything you need to know in order to communicate with ADAM. Next you will learn how to make ADAM get down to some serious work, and how to keep it busy for hours with only a few short program lines.

CHAPTER 8
MORE SmartBASIC

You have learned that ADAM can store vast amounts of information and perform simple arithmetic calculations accurately with lightning speed. ADAM also has the ability to make decisions and follow completely different sets of instructions according to the outcomes of these decisions. You can also program ADAM to repeat a set of instructions many times without retyping them each time it's run.

Branching

Programmers aren't referring to real trees with leaves, bark and limbs when they use the term *branching*. The branches about which they're talking are subsets of a program. Each branch is designed to perform a set of related instructions to accomplish a specific task. The branches join together to form the program just like the branches of a real tree join together to form the tree's trunk.

Have you ever watched a squirrel in a tree? ADAM's mind is like that squirrel. The squirrel runs up and down the tree's branches and could quite possibly spend some time on every branch of that tree. The squirrel, however, can be only one place at a time. That means that when you look up into the tree, you will probably see the squirrel staring back at you or scampering along one of the tree's limbs. Like this squirrel, ADAM's attention will be on only one of the branches of the program at any one time. Like our bushy-tailed example, ADAM's attention can jump from branch to branch, or run down an individual branch to the tree trunk (the main part of the program) and run back up another branch. Fortunately, ADAM doesn't have a tendency to gather nuts for the winter, but you need to gather the instructions required to tell ADAM to jump from one branch to another.

Suppose you want to begin by designing a program that contains just two branches. The first branch will give you instructions to get to the bank and the second will give you instructions to get to the post office. Begin by storing these instructions in two string variables, BANK$ and POST$.

```
]new

]10 bank$="Turn left on Ma
rket Street, go 6 blocks,
turn right and go 2 more b
locks."

]20 post$="Turn left on Mi
ssion Street and go 7 bloc
ks."
```

You can tell ADAM to print these directions by adding a line to the program that tells ADAM to print the string variables BANK$ and POST$, but how do you get ADAM to print just one set of directions? If you want to go to the bank, then you don't need ADAM to remind you how to get to the post office. In the previous sentence there are two short but powerful words that ADAM understands: "if" and "then." Notice how it makes the sentence conditional: "*If* you want . . . *then*. . . ." ADAM interprets "if" and "then" just as you do. ADAM will follow the instructions after the word "then" when the words following the word "if" are true. Let's go back to the example to see how this new command can apply:

```
]40 if place$="bank" then
print bank$: end

]50 if place$="post office
" then print post$: end
```

The colon before the END statement tells ADAM to expect another command. You can use several colons to squeeze many different commands onto the same program line. ADAM will follow the commands in sequence.

Getting back to the example, you have probably already noticed a slight problem. These two lines contain a string variable that hasn't been defined, namely PLACE$. This variable refers to the place to which you are requesting directions. You must have ADAM ask you where you plan to go before it can decide which set of directions to print. Do you remember how you did this in the last chapter? You have two choices, the

GET statement and the INPUT statement. The variable PLACE$ will need to assume the value of the word BANK or the phrase POST OFFICE. You can't use GET in this situation because, as you previously learned, this lets you type only a single character before ADAM continues to run the program. Therefore, use the INPUT statement to tell ADAM to stop and ask for your destination:

```
]30  input "Where do you w
ant to go?";place$

]run
```

ADAM will respond with:

```
Where do you want to go?
```

Suppose you want to go to the bank. Then type:

```
bank
```

ADAM will print:

```
Turn left on Market Street
, go 6 blocks, turn right
and go 2 more blocks.
```

Now ask for directions to the post office.

```
]run
```

ADAM again asks:

```
Where do you want to go?
```

This time answer

```
post office
```

ADAM then prints out the directions.

```
Turn left on Mission Stre
et and go 7 blocks.
```

The words IF and THEN must always be used together in the same program line. They can be used in many combinations, not only with PRINT, but also with GET, INPUT, SAVE, NEW, HOME, CLEAR, =, TAB,

and just about any command you can use in SmartBASIC. The IF . . . THEN combination causes *conditional branching,* that is, ADAM will perform any instructions in the branch following THEN, on the condition that whatever is between IF and THEN is true. If what follows IF is not true, the commands following THEN will be ignored. ADAM will not move along that branch. (Is that like saying ADAM won't go out on a limb?)

Let's go back to the example. Suppose that the information in the program is top secret. You would rather destroy the data than let it fall into unfriendly hands. Type:

```
]60 new
```

Now if BANK or POST OFFICE aren't input when ADAM asks the question, ADAM will ignore the branches in lines 40 and 50 (each of which tells ADAM to end), and continue to line 60, which tells ADAM to erase the program from working memory. If you do answer ADAM's query with one of the correct locations, you will receive the proper directions and the program will stop without being erased.

In addition to conditional branching, using the IF . . . THEN statements, you can instruct ADAM to branch unconditionally with another command, GOTO. GOTO, meaning simply "go to," is followed by a positive whole number or an expression that has a positive integral value. This value must equal a line number within the program. GOTO tells ADAM to skip to the line number that follows. For example, type:

```
]new
]10 print "SPRING"
]20 print "SUMMER"
]30 print "WINTER"
]40 print "FALL"
]50 goto 10
]run
```

Suddenly, you will see:

```
SPRING
SUMMER
WINTER
FALL
```

```
SPRING
SUMMER
WINTER
FALL
SPRING
SUMMER
WINTER
FALL
SPRING
SUMMER
WINTER
FALL
```

flashing endlessly across the screen. Each time ADAM gets to line 50, the GOTO statement sends it back to line 10. Then ADAM follows the program lines in numerical sequence until line 50 is reached again. ADAM will continue printing the names of the seasons again and again until mechanical failure forces it to stop. ADAM has been trapped in an *infinite loop* and will stay trapped until you provide the key to ADAM's prison. That key is <CONTROL>C, which will interrupt the program.

Let's try another example of GOTO. Type in the following.

```
]new
]10  print "ONE"
]20  print "TWO"
]30  goto 60
]40  print "THREE"
]50  print "FOUR"
]60  print "FIVE"
]70  end
]run
```

ADAM will respond:

```
ONE
TWO
FIVE
```

What happened to THREE and FOUR? ADAM branched around program lines 40 and 50, completely leaving out part of the program.

These two examples show the two most common errors that will occur when you use branching in your programs. The first, the infinite loop, is usually easy to see. However, if there aren't any PRINT statements within the loop, ADAM will seem to sit motionless; the screen won't change at all. The second type of problem is more difficult to find if your program is large. You may never realize that ADAM is ignoring part of your program. The best way to discover and solve these problems is to use the TRACE command (discussed in the last chapter).

Loops

Loops aren't necessarily bad. In fact, looping allows ADAM to perform many repetitive calculations with only a few program lines. After going around in circles for a finite number of times, ADAM will break out and continue with the remainder of the program. The most common way of creating a controlled loop is with a pair of words that must be used together. They are FOR and NEXT. The FOR must precede its associated NEXT in the program. Unlike IF and THEN, which must occur in the same program line, FOR and NEXT will occur on different lines. The loop encompasses both statements and every program line that falls between them.

Suppose you want ADAM to read a large volume of information in several DATA statements and to assign that information to several variables. If the number of variables and the amount of information is small, you should have no problem. But what if you need a hundred variables? A FOR . . . NEXT loop might be the best way to solve the problem. First create an array with the DIM statement. Then let ADAM feed the information from the DATA statements into that array. Here is an example:

```
]10 dim room$(100)
]20 for x = 1 to 100
]30 read room$(x)
]40 next x
]50 data . . .
]60 data . . .
      .
      .
      .
      .
and so forth
```

The FOR in line 20 begins by assigning a value of 1 to the numeric variable X. (You can't use a string variable with FOR.) Then ADAM goes to lines 30 and 40. When ADAM reaches line 40, it compares the value of the variable X to the number that follows the TO in the FOR expression (100 in this case). If the value of X is less than that number, ADAM will go back to the line with the FOR. At this point, ADAM increases the value of X by one and begins the cycle once more. The cycle will end after 100 loops when the value of X finally reaches 100.

You could do the same thing, but less efficiently, by combining IF . . . THEN with GOTO. Here's how:

```
]10  dim room$(100)
]15  x = 0
]20  x = x + 1
]30  read room$(x)
]40  if x = 100 then goto 60
]50  goto 20
]60  end
]70  data . . .
]80  data . . .
        .
        .
        .
        .
and so forth
```

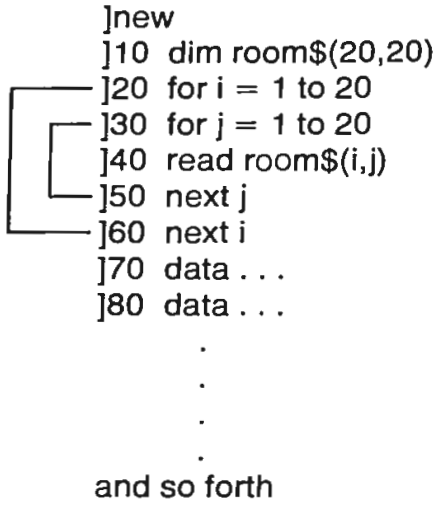
This program will produce the same results, but in a less efficient, more complicated manner. Line 20 in this program is not a misprint. The statement X = X + 1 is not an equation. This statement actually increases the value of the variable X by 1 each time ADAM sees the program line. It says, "The new value of X is the old value plus one." You can increase the increment value by changing the 1 to a larger number. That wouldn't do for this example, however.

Likewise, in the program on the previous page, you can increase the increment amount given with the FOR. For example, if you want to increase the incremental value to two, type:

```
]20  for x = 1 to 100 step 2
```

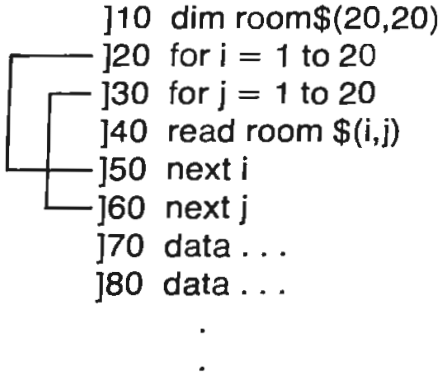
If you don't type the STEP # part, ADAM assumes that the step is one.

Suppose that you want to create a two-dimensional array and program ADAM to feed data into it. This requires the use of two FOR . . . NEXT loops, one inside the other. Since the inner loop is nested inside the outer loop, we refer to it as a *nested* loop. Here is what it looks like:



This program will cycle through the nested loop twenty times for each cycle of the outer loop. It will therefore read 20 times 20 (or 400) pieces of information. You can even have nested loops within other nested loops. There is no limit, but be careful; you can create a lot of work for ADAM with just a few program lines. For instance, if you had ten FOR . . . NEXT pairs, all nested within one another, and each only counted to ten, ADAM would have to go through ten times ten times ten . . . times ten (ten to the tenth power) cycles, or ten billion cycles. You might have time for a sandwich, a good night's sleep, and a European vacation before ADAM finished. Just remember, the total number of cycles is multiplied for nested loops and added for loops that are in different parts of the program.

Look at the lines to the left of the program in the previous illustration. Notice that the nested loop falls entirely within the outer loop. If you bracket associated pairs of FOR . . . NEXT statements, as in the illustration, you should never have one bracket crossing another. In the following example, the brackets do cross, indicating that the program is faulty.



.
.
and so forth

This program will result in an error when you try to run it. It's a common mistake that can easily be avoided just by drawing the brackets.

Subroutines

When you begin to write longer programs, you'll see certain patterns of statements that are used repeatedly. A good example is the set of FOR . . . NEXT loops that you just used to read an array of data. Wouldn't it be nice just to type in the set of steps once and refer to it when needed? That is exactly what SmartBASIC will let you do. The set of program steps is called a *subroutine*, since it is subordinate to the main program. Programmers usually put subroutines near the end of a main program after the END statement. Subroutines are small programs on their own, and are not accessed by ADAM unless you give a specific command. To access a subroutine, use GOSUB XX, where XX is a number, expression or variable that refers to a specific line number. Use the command RETURN at the end of the subroutine to tell ADAM to return to the main body of the program. ADAM remembers where it left the main part of the program, and returns to the program line immediately following the GOSUB statement.

Try this example:

```
]new

]10 print "SEE HOW I CAN
COUNT."

]20 gosub 100

]30 print "WASN'T THAT WO
NDERFUL?"

]40 end
]100 for i = 1 to 10

]110 print "      " i

]120 next i

]130 return

]run
```

ADAM should say:

```
SEE HOW I CAN COUNT.
        1
        2
        3
        4
        5
        6
        7
        8
        9
       10
WASN'T THAT WONDERFUL?
```

Subroutines can also be nested. That is, you can refer to a subroutine within another subroutine. For example, add these lines to the previous program:

```
]115 gosub 150
]150 print "   turn kick"
]160 return
```

Now run the program:

```
]run
SEE HOW I CAN COUNT.
        1
  turn kick
        2
  turn kick
        3
  turn kick
        4
  turn kick
        5
  turn kick
        6
  turn kick
        7
  turn kick
```

```
      8
  turn kick
      9
  turn kick
     10
  turn kick
WASN'T THAT WONDERFUL
```

Congratulations. You've just invented a new dance.

There is still one more command that you can use when you write subroutines into your programs. The command is POP. POP tells ADAM to forget the point in the program from which it branched out into the subroutine, and to return somewhere else when the RETURN statement is executed. That "somewhere else" is the line immediately following the very first GOSUB command in your program. Thus, if you have only one GOSUB command, POP has no effect. Let's use POP in the example of the nested subroutine.

```
]155  pop
]run
```

You should get something like:

```
SEE HOW I CAN COUNT.
          1
      turn kick
WASN'T THAT WONDERFUL?
```

Using POP too often can lead to very tangled programs, so use it sparingly. You also need to be careful about using GOTO in connection with subroutines. Use GOTO only to branch to program lines within the subroutine itself, or within the main program. Don't use GOTO to branch from the main program into a subroutine or vice versa. Such use inevitably will cause you headaches.

Multiple Branching

In SmartBASIC you can branch to many different places from the same program line. Two kinds of statements will allow you to do this: ON . . . GOSUB and ON . . . GOTO. These work much in the same fashion as GOSUB and GOTO, with which you are already familiar. They have the added feature of allowing you to vary the place to which you want to branch. Look at this example:

```
]new

]10 print "WHAT GRADE DID
YOU GET"

]20 print "IN HISTORY?"

]30 print

]40 print "TYPE THE CORRES
PONDING": print "NUMBER."

]50 print: print " A . .
1"

]60 print " B . . 2"

]70 print " C . . 3"

]80 print " D . . 4"

]90 print " E . . 5": pr
int

]100 get x

]110 on x gosub 200,300,4
00,500,600

]120 print: print "SAY HEL
LO TO YOUR TEACHER"

]130 end

]200 print "I'M VERY PROUD
OF YOU."

]210 return
```

```
]300 print "THAT'S GOOD."

]310  return

]400 print "YOU CAN DO BET
TER."

]410  return

]500 print "I'M DISAPPOINT
ED."

]510  return

]600  print "I HOPE YOU LIK
E SUMMER": print "SCHOOL."

]610  return

]run
```

ADAM should print:

```
WHAT GRADE DID YOU GET
IN HISTORY?

TYPE THE CORRESPONDING
NUMBER.

  A . . 1
  B . . 2
  C . . 3
  D . . 4
  E . . 5
```

Suppose you got an A. Now press the "1" key. ADAM should print:

```
I'M VERY PROUD OF YOU.

SAY HELLO TO YOUR TEACHER.
```

You will get a different message for each number that you input. This is another form of conditional branching. Each of the line numbers following GOSUB corresponds to a different number. The first line number corresponds to the number one, the second line number corresponds to the number two, the third line number . . . , and so forth. The variable or

expression between ON and GOSUB refers to one of those lone numbers with the corresponding number. For instance, if you were to input the number 2, ADAM assigns the value of 2 to the variable X. Since X equals 2, ADAM will choose the second line number of those listed in the ON . . . GOSUB statement (300).

The expression between ON and GOSUB must equal a positive whole number. It cannot be greater than the number of line numbers listed after GOSUB.

The ON . . . GOTO statement works just like ON . . . GOSUB, except that is doesn't branch to a line number. In the example, you could replace line 110 with:

```
]110  on x goto 200,300,4
00,500,600
```

and then replace all of the RETURN statements with GOTO 120. The program would run just as it did before.

A similarly useful command is ONERR GOTO. This branches in only one program line—that which follows the command. This will branch only if an error has been made, such as running out of data, dividing by zero, or taking the square root of a negative number. You can use this statement to branch to another part of the program and print your own error message to tell the user what the error was and how to correct it. At the end of that program segment use the RESUME statement to return to the main program. Actually, ONERR GOTO used with RESUME is very much like the combination of GOSUB and RETURN. Here is an example:

```
]new

]10  input "WHAT IS THE NUM
ERATOR?"; num

]20  input "WHAT IS THE DEN
OMINATOR?"; den

]30  print

]40  onerr goto 100

]50  ans = num/den

]60  print "THE ANSWER IS "
ans

]70  end
```

```
]100 print "I CAN'T DIVIDE
BY ZERO. ": print

]120 input "WHAT IS THE DEN
OMINATOR?"; den

]130 resume
```

Now try to divide 100 by 0, and then by 5.

```
]run

WHAT IS THE NUMERATOR?100
WHAT IS THE DENOMINATOR?0

I CAN'T DIVIDE BY ZERO.

WHAT IS THE DENOMINATOR?5

THE ANSWER IS 20
```

With this command you can make your programs much easier to use by predicting the kinds of errors that might occur and programming around them.

Arithmetic Functions

You've already used the four basic arithmetic functions—addition, subtraction, multiplication and division. Any pocket calculator can also do them. However, ADAM has a much larger repertoire that you haven't used yet. Let's start off with some simpler applications.

Within the confines of an IF . . . THEN statement, the equal sign has a different meaning than when you use it to assign values to variables. This different meaning is the one with which you usually associate the equal sign. It compares the expressions or values on either side of it and tests for equality. If they are equal, ADAM carries out the instructions after the THEN. In addition to the equal sign, you can use symbols that stand for the inequalities—less than, greater than and not equal to. Here is a table showing all of the possibilities:

=	is equal to
<>	is not equal to
>	is greater than
<	is less than
<=	is less than or equal to
>=	is greater than or equal to

You can use all of these with IF . . . THEN. When the statement is true, the instructions are followed. For instance:

```
]new
]10  x = 20
]20  y = 30
]30  if x = y then print "E
QUAL"
]40  if x <> y then print "
NOT EQUAL"
]50  if x < y then print "L
ESS THAN"
]60  if x > y then print "G
REATER THAN"
]70  if x <= y then print "
LESS THAN OR EQUAL TO"
]80  if x >= y then print "
GREATER THAN OR EQUAL TO"
]90  end
]run
```

ADAM will print:

```
NOT EQUAL
LESS THAN
LESS THAN OR EQUAL TO
```

Now type:

```
]10  x = 30
]run
```

ADAM will print:

```
EQUAL
LESS THAN OR EQUAL TO
GREATER THAN OR EQUAL TO
```

When you compare string variables, you can use only equal (=) and not equal (<>). The use of these functions is very straightforward and simple; you use them after IF and the meaning is the same as in common English.

Sometimes you need to make sure that a number doesn't have a fractional part. You might need to refer to a line number, as with GOTO or GOSUB, or you might need to refer to coordinates on a graph (as discussed in the next chapter). The INT function (short for "integer") will chop off the fractional part. For instance, INT(2.44) is equal to 2. INT(99.99) is equal to 99. You can also use INT to round numbers rather than just cut off their fractional tails. Just add one half (0.5) to the number in the parentheses. Try to round off the numbers 2.4, 5.6, 99.99, and 65.49:

```
]print int(2.4+0.5)
2

]print int(5.6+0.5)
6

]print int(99.99+0.5)
100

print int(65.49+0.5)
65
```

As you can see, ADAM does a fine job with this. Sometimes, however, ADAM is not completely accurate. Type:

```
]new
]10  print int(100)
]20  print int(100.0)
]30  print int(100.00)
]40  print int(100.000)
]run
```

ADAM will print:

```
100
100
99
100
```

When you put two decimal places after a number, ADAM thinks that the number is slightly smaller than it actually is. To measure that difference, type:

```
]print 100 - 100.00
```

The answer that you get will not be zero. In fact, it is:

```
2.98023224 E-08
```

This is very close to zero, but it could be a problem, especially when you compare two numbers to see if they are equal. This bug apparently occurs only when you use two decimal places, and for numbers less than 1,025. If you can, try to stick with one decimal place or three or more.

Not only can ADAM remove the fractional part of a number, it can also remove the "sign" information (the part of the number that tells whether it is negative or positive). ADAM can tell you the number's *absolute* value. The function has the form ABS(XX), where XX is any numerical value—an expression, a numeric variable or a number. This is most often used to prevent errors when used with the SQR (square root) function. If you try to take the square root of a negative value, ADAM will give you an error message. To get around this, take the absolute value before taking the square root:

```
]print sqr(abs(-4))
```

When you type this, ADAM will print:

```
2
```

This prevents the error message, but it isn't exactly correct. Negative numbers actually have square roots, called *imaginary* numbers. Though you will probably never need to use imaginary numbers, they do indeed exist, and an I or a J is placed after the number so you'll know that it's imaginary. ADAM cannot deal with these numbers directly. With the help of one more function, however, ADAM can. This new function is the SGN or "sign" command. SGN(XX) will tell you if the value XX is positive, negative or zero. It discards the information that the ABS function saved.

ABS and SGN are complementary. Each keeps the information that the other throws away. You can therefore break any number into two components—the magnitude and the sign. With this in mind, let's go back to our square root problem. First, break the number into components. Then take the square root of the absolute value. If the sign is positive, you have the answer. If the sign is negative, however, you merely

need to put the letter i after the square root of the absolute value. Here is a subroutine that will take the square root of any value, positive or negative, and print the square root:

```
]500 y = sqr(abs(x))

]510  z = sgn(x)

]520  if z = −1 then print
"THE SQUARE ROOT IS " y "
i": goto 540

]530  print "THE SQUARE ROO
T IS " y

]540  return
```

Suppose you write the following program to access the subroutine:

```
]10  input "GIVE ME A NUMBE
R: "; x

]20  print

]30  gosub 500

]40 end
```

Now find the square roots of 25, 36, −16, and −102400:

```
]run
GIVE ME A NUMBER: 25
THE SQUARE ROOT IS 5
]run
GIVE ME A NUMBER: 36
THE SQUARE ROOT IS 6
]run
GIVE ME A NUMBER: −16
THE SQUARE ROOT IS 4i
]run
GIVE ME A NUMBER: −102400
THE SQUARE ROOT IS 320i
```

The last answer makes one wonder whether BMWs are imaginary. You might never hear about an imaginary number again, but the next time someone tells you that you can't take the square root of a negative number, you'll know something he doesn't.

SGN is also useful for branching. Suppose you write a financial program that will tell you to go see a bankruptcy lawyer. When your net worth is less than zero and you're bankrupt, the amount by which your liabilities exceed your assets is not a determining factor. You can then use the SGN function to test your net worth at any time.

ADAM doesn't think of numbers and characters the same way you do. ADAM converts everything into a special set of numbers called ASCII (pronounced "ask-key") codes. There are about 100 of them and each one corresponds to one of the characters on the keyboard (including the various <CONTROL> LETTER combinations). You will rarely need to concern yourself with this, but ADAM can tell you the ASCII codes if you ever do. That way you won't have to try to get a copy of the conversion table and do your own translation. To find the ASCII code that corresponds to a particular character, in this case the letter M, type:

```
]print ASC("M")
```

ADAM should print:

```
77
```

If you want to find the character that corresponds to a particular ASCII code, use the CHR$ command. For example, the character corresponding to the ASCII code 42 is an asterisk.

```
]print chr$(42)
*
```

That's about all you need to know for now about ASCII codes. Just remember these two commands if you ever need to translate back and forth.

ADAM also contains a random number generator. This is important when you want to make ADAM less predictable in games or educational "flash card" programs. The function has the format RND(XX), where XX is a numerical expression or variable. Mathematicians call the XX part of the command the kernel. ADAM is supposed to generate a different stream of random numbers for each kernel, but it won't. In fact, ADAM will generate the same stream of numbers each time you run the program. Each number in the stream may be individually random, but the stream itself can be predicted. You can get around this flaw by creating a loop that generates random numbers at the beginning of the program but doesn't print them. You can then generate random numbers by truncating different qualities of numbers from the beginning of the stream of numbers. To illustrate:

```
]new
]10  input "WHAT IS THE KER
NEL? ";k:?
]20  for x = 1 to 5
]30  print rnd(k)
]40  next x
]50  end
```

ADAM will print out the same stream of random numbers no matter which positive value you input.

```
]run
WHAT IS THE KERNEL? 33
.732004777
.425420012
.0831831896
.705190617
.69693043
```

Now try a different kernel:

```
]run
WHAT IS THE KERNEL? 250
.732004777
.425420012
.0831831896
.705190617
.69693043
```

The values are the same, although they shouldn't be. Now try this program:

```
]new
]10  input "WHAT IS THE KER
NEL? ";k:?
]11  for y = 1 to k
]12  z = rnd(1)
```

```
]13  next y
]20  for x = 1 to 5
]30  print rnd(1)
]40  next x
]50  end
```

Now run the new program with kernels of 1, 3, and 5:

```
]run
WHAT IS THE KERNEL? 1
 .425420012
 .0831831896
 .705190617
 .69693043
>>>  .650009534
]run
WHAT IS THE KERNEL? 3
 .705190617
 .69693043
>>>  .650009534
 .141860894
 .720451122
]run
WHAT IS THE KERNEL? 5
>>>  .650009534
 .141860894
 .720451122
 .137060179
 .963411333
```

The "arrow," >>>, points out the same number in each of the runs. Notice that as you increase the size of the kernel, the number of random numbers ahead of it is decreased by the kernel's increase. This shows you that the same stream of numbers is actually being generated each

time. You get unique streams by discarding a certain number of values from the stream's beginning. The major disadvantage of this method is the time lag that a large kernel will cause as ADAM performs the first FOR . . . NEXT loop. You haven't got much of a choice, though, if you need a stream of random numbers.

You may have noticed that the random numbers are always fractional, and they all fall between zero and one. In order to generate streams of larger numbers, simply multiply the random numbers by a factor equal to the largest possible value, and then round off the fractional part with the INT function. For example, suppose you want to generate a stream of random numbers that can range from 0 to 1000. Just change the previous program with:

```
]30 print int(rnd(1)*1000+
0.5)
```

If you use a kernel of 5, you will get this stream:

```
650
142
720
137
963
```

If you don't want zero as your lower boundary, just add the number you want that boundary to be. The number by which you multiply the RND function will be the difference between the upper and lower boundaries. The one half is added so that you will get rounded values instead of truncated values. These steps will show you how:

```
]22 input "LOWER BOUNDARY?
 ";lower

]24 input "UPPER BOUNDARY?
 ";upper

]26 range = upper - lower

]30 print int(rnd(1)*range
+0.5)+lower
```

In this example the lower boundary should be less than the upper boundary. Both boundaries can be negative numbers, however.

We will now discuss the calculation of logarithms and exponential values. If you don't understand them you may want to skip to the section on defining your own functions.

An exponential value is a number raised to the power of another number. The exponent with which you are most familiar is 2. When a number is raised to the power of 2, we say that it is squared. When the exponent is 3, the number is cubed. There aren't any common terms for other exponents. The inverse of raising a number to a power is taking that number's root. You've already learned how to tell ADAM to find the square root of a number.

A related concept is that of the logarithm. The three are related as follows:

$2^3 = 8$ (exponentiation—2 raised to the power of 3)

$\log_2 8 = 3$ (taking the logarithm of 8 with a base of 2)

$\sqrt[3]{8} = 2$ (finding the third (cube) root of 8)

ADAM already has functions that will calculate any powers, find square roots of positive numbers, and take the logarithms (using a base of e, always lower-case) of any positive number. You can create your own functions to calculate logarithms with other bases and to find other roots.

To raise a number to a power, just type an up-pointing arrow between the base and the exponent. The arrow (not one of the cursor keys) is located just above the return key. The base and exponent can be any numeric values. Type:

```
]new
]x = 4
]y = 5
]print x^y
```

ADAM will print:

```
1024
```

which is exactly the number you get if you multiply 4 (the base) by itself 5 (the exponent) times. You could do this yourself. Now try a more difficult example:

```
]print 2.3 ^ 6.78
283.474977
```

You might have some difficulty calculating an answer like 283.474977. That's why you have ADAM, so you won't ever have to.

You've been exposed to the SQR function earlier, so let's go on to the logarithm function, LOG(XX), where XX is any positive numeric value. When you use this function, ADAM assumes that the base is a strange but useful number called e. e is a *transcendental* number, like pi (used to find the areas of circles). e is widely used in calculus and in the mathematical sciences. It is approximately equal to 2.718282.

One of the tricks of logarithms allows you to calculate quickly the factor that will convert logarithms with one base to another. Here are a formula and a table that you can use with ADAM:

Let alpha be the new base.

Let X be the number whose logarithm you are calculating.

Then:

$$\log_{alpha}(x) = \log_e(x) / \log_e(alpha)$$

LOGARITHM CONVERSION TABLE

BASE	$1/\log_e(BASE)$
2	1.442695
3	.910239
4	.721348
10	.434294

To use this table, multiply the value of LOG(XX) by the number in the right-hand column for the base you want to use.

Defining Your Own Functions

SmartBASIC allows you to create your own functions. You can do this with the DEF FN command followed by a name that you give to the function and an expression or equation which the function will represent. The function acts almost like a tiny subroutine. For instance, let's create a logarithmic function having a base of 10. We'll call it LG and use it to stand for the equation listed above the logarithm conversion chart.

```
]new
]100  def fn lg(x) = log(x)/log(10)
```

Now type:

```
]110 print fn lg(100)
]200 end
]run
```

ADAM should print:

```
2
```

which is the logarithm of 100 with the base of 10. The X in the statement is just a dummy variable—it doesn't have any particular value.

Let's define another one, called RT, which will calculate the cube root of a positive number.

```
]110 def fn rt(x)=2.718282
^(log(x)/3)
```

The 2.718282 is the value of e. You can change this to find any root by replacing the 3 with that number. For example, substituting a 4 will make the function find the fourth root of a number.

Let's create a simple function that rounds numbers. We'll call it ROUND.

```
]120 def fn round(x)=int(x
+0.5)
```

Here are some rules to remember when you create functions.

1. ADAM looks only at the first two letters of the function name. If later in the program you define another function with the same two beginning letters, ADAM will ignore the first one.
2. You can only define functions in the program mode, never in the immediate mode. However, a user-defined function that has been created in a program can be used in the immediate mode before NEW, CLEAR or LOAD commands are given.
3. You must make sure that all of the values in the DEF FN statement have been defined before invoking the function.
4. You can use previously defined functions in the DEF FN statement when defining a new function. The new function can't be used in the definition, of course.
5. The entire DEF FN statement must appear on a single program line.

ADAM already knows one trigonometric function—the sine function. You may ask ADAM for the sine of a number by typing:

```
]print sin(X)
```

Since ADAM doesn't have an inverse sine function (arcsine), here is a way to make your own:

```
]100 def fn isin(x)=x+(x^3
/6+x^5*3/40+x^7*15/336)
```

This approximates the first few terms of an exponential series that calculates the inverse sine of a number. Next we will similarly define the itan (inverse tangent) function:

```
]105 def fn itan(x)=x-x^3/
3+x^5/5-x^7/7
```

You can now use these functions, along with the ones ADAM already knows, to calculate the other trigonometric functions, such as icos (inverse cosine), tan (tangent), sec (secant), csc (cosecant), and cot (cotangent):

```
]110 def fn icos(x)=3.1415
926/2-isin(x)

]120 def fn tan(x)=sin(x)/
cos(x)

]130 def fn sec(x)=1/cos(x
)

]140 def fn csc(x)=1/sin(x
)

]150 def fn cot(x)=cos(x)/
sin(x)
```

The cotangent function can also be defined as follows:

```
]150 def fn cot(x)=1/tan(x
)
```

You are now prepared to solve just about any trigonometric problems you come across. This was the toughest chapter in the book, since you covered a lot of material very quickly. Now get ready to explore the fascinating world of graphics and games.

CHAPTER 9
GRAPHICS

If you've played Buck Rogers: Planet of Zoom or any other games on ADAM, you may already be wondering how programmers create games. What do these games have in common with the simple programs you've learned to write? Games are just programs that provide a great deal of interaction with the player. Arcade style games use one thing that you have yet to learn, however. They use the graphics capabilities of the computer to create the colorful, changing images of the television screen. At first you may not think that you can create anything so complex, but with ADAM you can easily construct shapes and figures on the screen and learn how to move them around to create your own games or works of art.

The term "graphics" comes from a Greek word meaning "to draw" or "to write." You can see a form of the word in "telegraph" (to "write" from afar) and "phonograph" (to "write" sound). With ADAM's graphics, you can draw directly on the screen with light that is controlled by the commands you give ADAM.

So that you can tell ADAM precisely where to draw a line or a point, ADAM invisibly divides the screen into tiny rectangles, each with its own name. ADAM automatically creates an array with each member of the array corresponding to a particular rectangle on the screen. You do have something to say in this matter, however, for ADAM has two graphics modes—*low resolution* and *high resolution*. The main differences between the two are the amount of memory required to maintain the array, the size and number of rectangles, and the commands that must be used. The high resolution graphics mode divides the screen into a larger number of smaller rectangles than the low resolution mode. High resolution demands more memory to maintain than low resolution. It therefore leaves less working memory for you to use for programming, while allowing you to create smoother, more intricate shapes and designs. We'll begin with the simpler of the two, the low-resolution mode.

Low-Resolution Graphics

In this mode, ADAM divides the screen into 1600 rectangles, organized into 40 columns and 40 rows. Each of these is called a *coordinate*. The coordinate is described by two whole numbers or numeric variables separated by a comma. The first number or variable refers to the column and the second to the row, and they can range from 0 to 39. For example, the coordinate 20,30 refers to the rectangle in column 20 and row 30. Once you understand this system, you'll have no trouble telling ADAM what you want.

Begin by telling ADAM to enter the low-resolution graphics mode by typing:

```
]gr
```

The GR command is simply an abbreviation for "graphics." Suddenly the screen will go blank, but any programs you have in ADAM's working memory will still be there. If the programs are very long, save them first, since the remaining amount of memory might not be sufficient.

The screen is now composed of the invisible grid and four rows at the bottom of the screen where you can type program lines and commands. Notice that the cursor has moved there already, awaiting your next command. At this point you might want to select the color with which you want to draw. If you have a black-and-white TV, the choice of different colors will result in various shades of grey. If you are using a color set, however, you can take full advantage of the choice. ADAM allows you to choose any of 16 different colors. You can refer to each one by a number, or color code, according to the following table:

COLOR CODE TABLE

0 — black
1 — magenta
2 — dark blue
3 — dark red
4 — dark green
5 — grey—1
6 — medium green
7 — light blue
8 — light yellow
9 — medium red
10 — grey—2

11 — light red
12 — light green
13 — light yellow
14 — cyan
15 — white

The actual shades of these colors will vary with individual television sets and color setting. For now we'll stick with these names. Let's choose light red to start.

```
]color = 11
```

Now you can see if ADAM followed your command by typing:

```
]plot 20,30
]plot 20,20
]plot 20,10
```

You should now see three small, light red rectangles appear in a vertical line near the middle of the screen. The PLOT command tells ADAM to change the color of the rectangles to the last color that you told ADAM to use. You can use COLOR and PLOT in programs just like the commands you already know. Try this simple program:

```
]new
]10  color = 15
]20  for count = 10 to 30
]30  plot 20,count
]40  next count
]50  end
]run
```

ADAM should draw a vertical white line over the points that you plotted previously. Notice that the colors don't mix; they remain pure. When a new point is plotted over an old one, the color always changes to the last color told to ADAM.

You can use another command to do exactly what the last program did. This is the VLIN command, and with it you can tell ADAM to draw a vertical line. For example, you can replace the old program with a new one.

```
]new
]10 color = 7
]20 vlin 10,30 at 20
]30 end
]run
```

The white line should suddenly turn light blue (color = 7). ADAM has just drawn this new blue line over the white one. The VLIN command told ADAM to draw a vertical line from row 10 to row 30 on column 20. Now type:

```
]15 for count = 1 to 10
]16 line = 4 * count - 4
]20 vlin 0,39 at line
]25 next count
]list
```

ADAM will list your program steps:

```
]10 color = 7
]15 for count = 1 to 10
]16 line = 4 * count - 4
]20 vlin 0,39 at line
]25 next count
]30 end
```

Be sure to use <CONTROL>S to stop ADAM from scrolling the first few lines past the text region of the screen. Then press <CONTROL>S to see the last few lines. Now run the program. ADAM will create 10 vertical lines, beginning with column 0 and ending with column 36. It will look like a forest after a fire.

Now how can you turn this into a grid with horizontal lines as well as vertical ones? You guessed it: use the HLIN command. This works exactly like VLIN except that the lines are drawn horizontally. Try this:

```
]list 20
```

ADAM responds with:

```
]20 vlin 0,39 at line
```

Let's edit this line by changing the VLIN to HLIN using the shortcut method you previously learned. Move the cursor until it is under the V in VLIN, replace the V with an H, scroll the cursor to the end of the line, and hit <RETURN>. If you list line 20, it should now read:

```
]20 hlin 0,39 at line
```

When you run the program, ADAM will create a set of horizontal lines and complete the grid.

Before you move on to new material, try this little program:

```
]new
]10  for i = 0 to 15
]20  color = i
]30  vlin 0,39 at i + 5
]40  next i
]50  end
```

You will now see the entire spectrum of colors.

There is only one more statement that you need to know when using low-resolution graphics. It is called the SCRN function. Like PLOT, you need to follow it with two numbers or numeric variables, separated by a comma, so that ADAM will know which rectangle to reference. Unlike PLOT, you need to put parentheses around the numbers or variables. SCRN instructs ADAM to look at the designated rectangle and tell you its color. (ADAM will give you a color code; you have to look it up on the table yourself, unless you write a short conversion program.) For instance, type:

```
]print scrn (20,20)
```

and ADAM will print:

```
7
```

This corresponds to light blue, the color of that rectangle. This is a very subtle function, and you can use it for some advanced programming.

You now know everything about low-resolution graphics. When you want to go back to the text mode, simply type in the command TEXT and ADAM will return you to the standard text mode that greets you when you first load SmartBASIC. The screen will be blank except for the cursor. This is also a fast way to erase your diagrams. Just type:

```
]text
]gr
```

and you'll be back in the low-resolution graphics mode with a solid black screen. Try this, and then type in the following program:

```
]new
]10  x = int(rnd(1)*37)+1
]20  y = int(rnd(1)*37)+1
]30  color = 11
]35  rem THIS PLOTS THE
```

```
]36 rem   CENTER
]40 plot x,y
]50 color = 13
]55 rem THIS PLOTS THE
]56 rem   PETALS
]60 plot x+1,y
]70 plot x-1,y
]80 plot x,y+1
]90 plot x,y-1
]100 goto 10
]run
```

Let's hope you like flowers. (That's what they're supposed to be.) ADAM should begin to draw a field of sunflowers and will continue drawing sunflowers until you stop the process by typing <CONTROL>C. Notice that this locks ADAM into one of those nasty infinite loops. The reason for multiplying the RND(1) by 37 is to generate a set of random numbers ranging from 0 to 37, since RND(1) can range from 0 to 1. We used the INT function, because ADAM can't accept numbers with fractions as coordinates. The reason for a range of 37 numbers is to allow room for the flowers' petals. Adding 1 to the coordinate (lines 10 and 20) shifts the range from 0 – 37 to 1 – 38. This leaves room for petals along all four borders of the graph region.

You can create your own flowers, dogs, fish, trees, and so forth. Use your imagination.

High-Resolution Graphics

For detailed drawing, you must use the high-resolution graphics mode. In this mode ADAM has 40,960 tiny rectangles that you can access. You will have 256 columns and 160 rows at your disposal. The rectangles will be so small that they will actually appear only as points of light. And if that is not enough, ADAM has a second high-resolution mode that will incorporate the four-line text region at the bottom of the screen into the graphics region, giving you 32 more rows to access. It is in this second high-resolution graphics mode that most of ADAM's arcade-style games are played.

The commands that you use to get into the two modes are HGR and HGR2, respectively. Before you enter one of them, review the flower program in the previous section. Now you will learn how to alter it so that it

will work in the high-resolution modes. The high-resolution commands equivalent to COLOR and PLOT are HCOLOR and HPLOT. They work in very nearly the same way. The color codes, though, are different. They are:

HIGH-RESOLUTION
COLOR CODE TABLE

0 — black 1
1 — green
2 — violet
3 — white 1
4 — black 2
5 — orange
6 — blue
7 — white 2
8 — brown
9 — dark blue
10 — grey
11 — pink
12 — dark green
13 — yellow
14 — aqua
15 — magenta

Now edit the flower program, changing COLOR to HCOLOR, PLOT to HPLOT, and the range of possible values in the statements containing the RND function. It will be much easier to do the editing while you are in the text mode. The resulting program should be:

```
]10  x = int(rnd(1)*253)+1
]20  y = int(rnd(1)*157)+1
]30  hcolor = 11
]35  rem   THIS PLOTS THE
]36  rem   CENTER
]40  hplot x,y
]50  hcolor = 13
]55  rem   THIS PLOTS THE
]56  rem   PETALS
]60  hplot x+1,y
]70  hplot x-1,y
]80  hplot x,y+1
]90  hplot x,y-1
]100  goto 10
```

Now type:

```
]hgr
```

The screen will go black. You are now in the first high-resolution mode. Run this program and compare it with low-resolution graphics. The difference is astounding. Type <CONTROL>C when you want ADAM to stop.

You should be aware that colors don't always remain pure in the high-resolution modes. You often get a result similar to the one you get when you wash a new pair of blue jeans with a white shirt—the color may "bleed" into the next tiny rectangle.

There are no commands in these modes analogous to the HLIN, VLIN and SCRN functions that you used in the low-resolution mode. To create lines, use the HPLOT command in a slightly different way. You already know how to plot a single point. For instance, to plot a white dot near the center of the screen, simply type:

```
]hcolor = 3
]hplot 140,80
```

If you type

```
]hplot 140,159 to 140,0
```

ADAM will plot a vertical white line near the center of the screen. You only need to provide the end points of any line segment and ADAM will draw the line between those two points. To create a vertical line, keep the first numbers of the two pairs of coordinates the same. Keeping the second numbers the same will create a horizontal line. Unlike the low-resolution mode, the lines don't have to be vertical or horizontal. Try drawing the following diagonal line:

```
]hplot 0,0 to 255,159
```

Try the following program, which will draw an interesting picture using diagonal lines:

```
]text
]new
]10  hcolor = 5
]20  for count1 = 1 to 80
]30  x = count1 + 40
]40  hplot 140,80 to 180,x
]50  next count1
]60  hcolor = 6
```

```
]70  for count2 = 1 to 80
]80  y = 180 - count2
]90  hplot 140,80 to y,120
]100  next count2
]110  hcolor = 8
]120  for count3 = 1 to 80
]130  z = 120 - count3
]140  hplot 140,80 to 100,z
]150  next count3
]160  hcolor = 14
]170  for count4 = 1 to 80
]180  a = 100 + count4
]190  hplot 140,80 to a,40
]200  next count4
]210  end
]hgr
```

This should result in a colored square composed of diagonal lines appearing near the center of your screen. Create some of your own images until you feel comfortable using ADAM's graphics capabilities.

Game Controllers

ADAM comes with two game controllers. They plug into the right side of the console and provide a convenient way for you to input cursor control information rapidly. When you press the joystick on top of the controller from side to side, you are sending a signal to ADAM. That signal can have values ranging from 0 to 255. (Notice that this corresponds directly to the column numbering in the high-resolution graphics mode.)

You can determine that value with the PDL command. Since you have two controllers, ADAM refers to the first controller by putting a zero in parentheses after the PDL function. A one in parentheses refers to the second game controller. Type:

```
]print pdl(0)
```

ADAM will print the number corresponding to the position of the first joystick. Type:

```
]print pdl(1)
```

ADAM will print the number corresponding to the position of the second joystick. This is the function ADAM used to determine if the joystick is in the right position to blast an alien spacecraft when you play the Buck Rogers game.

High-Resolution Shapes

ADAM will also allow you to create shapes and store these images for future use. Creating them requires information that is not yet available. This information concerns the codes necessary for direct access to specific memory locations. The concept behind their creation is complex. With any luck, Coleco will soon publish the necessary information. Although you won't learn how to create a "shape table" here, you might want to be aware of some of the important points. In the future you may want to attempt to use this feature, especially if you ever want to create your own games.

You could begin by defining a set of shapes—triangles, squares, trapezoids, and so forth—and then storing those shapes in ADAM's memory. You could then instruct ADAM to draw those shapes in any size (with the SCALE command) and in any orientation (with the ROT command). In game applications, you might define shapes that you want to use to represent space ships and asteroids. You could then manipulate them so that they appear to move. Getting closer and larger, then farther and smaller, rotating clockwise, then counterclockwise.

Youngsters of all ages are learning to use these techniques to design games. Some make thousands of dollars doing it. But they're not all geniuses; if you advance one step at a time, you too will learn how to make ADAM perform all sorts of wonderful and interesting tasks. It just takes patience and practice.

CHAPTER 10
PROGRAMS

To program, or not to program—for years, ever since the first home computer was invented in the garage of a California hobbyist, computer buffs have debated whether ordinary users of machines like ADAM should learn to program. Some people argue that in the not too distant future computer users will be able to buy whatever program they need right off the shelf. Indeed, thousands of computer programs are already commercially available. Their applications range from word processing and filling out tax forms to keeping tabs on the family budget. In a sense, buying ready to go software has become as easy as going to the local computer store.

However, there is another school of thought that says that to understand computers thoroughly, one must program. It is only by learning to program, the argument goes, that a user can grasp the true computing power of a machine like ADAM. It is only by reading through a program step by step, pursuing the logic, and following the reasoning, that a beginning computer user can understand how a computer thinks.

It is in this spirit that we offer you a group of elementary programs in SmartBASIC. Our feeling is that by providing some already written, ready to go programs that do some pretty amazing things, we achieve two goals. First, we show you how useful SmartBASIC can be when it is applied in a program. More important, once you work with our programs and see how practical SmartBASIC and ADAM can be, you will probably become as enthusiastic as we are about ADAM and will want to write your own programs to do the specific jobs that you need done. You'll soon learn that SmartBASIC can be used for all sorts of applications—for the home, office or school.

To get you started, we've provided eight different programs, each designed to do a specific job much more quickly than you could on your own. One program, COMPOUND INTEREST, will automatically compute the interest earned on an initial amount of money deposited in a bank. You can use this to watch your money grow over time. Another program, COLLEGE TUITION, will automatically figure how much money you

should be putting aside each year to save for your child's college education. Other programs compute monthly payments on a home mortgage, convert foreign currencies to United States dollars, balance your checkbook, and teach your children arithmetic. The only real limit to how you can use SmartBASIC is the breadth of your imagination.

Just as important is the fact that all of the programs that follow have been written specifically for ADAM. Unlike generic programs that must be adapted to work on a given machine, these programs will work immediately on ADAM. You don't have to make any changes at all to get started. All you have to do is type in the programs exactly as written, type in RUN, and away you go—you're computing. It's that easy!

Prepare yourself for one of the most exciting aspects of computers by brushing up on your knowledge of the SmartBASIC commands outlined in the preceding chapters. You might want to review some of the brief programs we provided earlier before plunging into the more detailed programs ahead, but don't waste too much time reviewing. Chances are you'll want to put ADAM to work almost immediately. With any luck at all, you'll be programming ADAM like a professional before long, and in doing so you will have opened up a whole new world of computing that will educate you, free you from onerous chores and computations, and even, in the case of financial management programs, put money in your pocket.

How to Use the Programs

Each of the eight programs is described in four sections.

The first, Purpose, tells you precisely what the program does and how you can use it to your best advantage.

The second section, called How to Run It, explains what happens when you run the program and provides descriptions of information you must input as the program is executed. It's important that you give your responses precisely as described, taking special care to use capital letters and correct punctuation.

The third section, entitled SmartBASIC Listing, provides a printout of the SmartBASIC program itself. You'll recall from earlier chapters that a SmartBASIC program is nothing more than a list of instructions that tells ADAM what to do. It's crucial, therefore, that you type in the program exactly as written.

In the fourth section, entitled Sample Output, we show you what you should see on your screen once you type RUN to execute the program. You can check if your own program is running correctly by comparing the output on your screen with the sample output provided in the text.

If you're just getting started with programming, it's probably best to read through all four parts of a program before setting down to type in the individual program lines. Once you understand what the program does and what you have to do to make it run correctly, enter the NEW command to erase any program that may currently be in ADAM's memory. Then, using the SmartBASIC Listing section as a guide, begin typing in the program, taking extra care to get all the letters and punctuation correct. Once you've finished typing in the text, immediately store the program in ADAM's memory so that you don't lose it if the power goes out.

Before running the program, it probably pays to get a listing of it using SmartBASIC's LIST command. This way, you can be sure that what you entered is precisely what appears in the book. Don't forget: Use capital letters where they are requested, and try not to confuse the letter I with the number 1 or the zero with the letter O.

Now run the program. Your result should be pretty much the same as what appears in the Sample Output section of the text. If that's the case, congratulations! You've just written and executed your first applications program in SmartBASIC. If you get an error message—or if the output is different from ours—try reading through the program line by line, looking for spelling errors, transposed words, incomplete commands or other typos. Once again, make sure you've used all proper punctuation and have capital letters when called for.

If your program still doesn't work correctly, try running through the reasoning of the program out loud. Consider each line of the program individually, making sure that you understand what the line is telling ADAM to do. Are you sending ADAM the right message? If not, rewrite the line. You might try making a flow chart listing the processes ADAM goes through in the course of your program.

If that doesn't work, try using the TRACE/NO TRACE command described in the sections on SmartBASIC. The TRACE command shows the order of the lines ADAM reads as it executes your text.

If you still can't get your program to work properly, don't despair. There is still a solution. Try inserting a PRINT statement midway in your program instructing ADAM to print out the value of all variables. If ADAM prints the correct values, you know that your bug lies below the PRINT statement. If the values are wrong, you know the problem lies above the PRINT statement. You then can insert another PRINT statement to home

in on your problem. Eventually, you will be able to focus your debugging efforts on a small section—say, five lines—where the error seems to be coming from. From that point, it should be easy to pinpoint the problem and to root out the rascal. Removing the kinks from even a complicated program doesn't have to be hard.

However, since all of the following programs have been tested to work correctly on ADAM, the first piece of advice is most important: Type the programs in correctly and they will work from the start. This holds true not only for our programs but for any others you buy or borrow from a friend. Follow the text verbatim and it should work. If you do this, you'll spare yourself a lot of debugging time, and you'll have more time to enjoy ADAM.

LOAN AMORTIZATION

Purpose: In these days of rising prices and volatile interest rates, it's important to keep track of expenses. For families and individuals alike, one of the most common expenses is the monthly payment on an installment loan—a home mortgage, a note on a boat, a car payment or some other type of long-term I.O.U. Figuring out the monthly payment on a loan can be a complicated affair, until now the preserve of bankers and accountants. ADAM can automatically compute the monthly payment, including principal and interest, needed to pay off most common consumer loans. The program has been written to provide the monthly payment on a car loan or mortgage, but it can be used to figure out the monthly payment on any type of obligation—a student loan, a home improvement loan, even a debt between friends.

How to Run It: ADAM will ask you a series of questions regarding the terms of your loan—the amount you would like to borrow, the down payment you wish to make and the annual interest rate you wish to pay. Answer the questions exactly as you would if you were applying for a loan from a neighborhood savings association. To be sure the program runs properly, follow these guidelines:

1) Don't put dollar signs or commas in numbers when ADAM asks for dollar amounts. For example, the amount $10,000 should be typed into ADAM as 10000, the amount $25,000 as 25000.

2) When ADAM asks for the annual interest rate you'd like to pay on your loan, type in a whole number, not a percentage or fraction. For example, when ADAM asks for the interest rate on a 12 percent loan, it should be described simply as 12, not 12/100 or .12. It's all right to have a decimal in a percentge, though, such as 13.8 percent.

3) In the output, ADAM will tell you the monthly interest rate you're paying on your outstanding debt. Here decimals are used to express whole percentages. If the monthly interest rate is 3 percent, ADAM would type this as .03. A monthly rate of 12 percent would be characterized as .12.

COMPOUND INTEREST

Purpose: Mutual funds, money market accounts, Treasury bills, certificates of deposit—today one has only to open the financial pages to see the myriad investment options available. Each choice brings with it different minimum investment requirements, a different interest rate earned, and different compounding periods (compounding refers to the fact that investors can earn interest on their interest). How is a saver to choose among this confusing array of investment vehicles? One answer is the program below. When you input information about the terms of your investment, ADAM will automatically show how your money grows over time as you earn interest. By running the program repeatedly, you can easily compare different investments.

How to Run It: ADAM will ask you a series of questions regarding the investment you're considering, such as the amount of money invested, the interest you will earn, how often your interest will compound (once a year, semiannually, every day), and how far out in time you would like calculations. Follow these guidelines to make the program work as meant:

1) Type in dollar amounts as full numbers without dollar signs or commas. For example, type 10000 for $10,000, 20000 for $20,000.

2) When ADAM asks you how much interest you will earn on your investment, type in the full number with a percentage sign. For example, type 10 for 10 percent. It's acceptable to have a decimal point in your interest rate, i.e., 12.3 percent.

3) When ADAM asks how many times a year you'd like your interest to compound, use a number, not words. For example, semiannual interest would be characterized as 2, monthly compounding as 12.

COLLEGE TUITION

Purpose: For many couples, financing a child's higher level education can be as difficult as paying high doctor bills or a huge mortgage. It's enough to break the budget even of wealthy families. Fortunately, there is an answer: financial planning. By putting aside money now and saving each year, it's possible to accumulate a tuition pool large enough to pay off those big school bills when your child goes to college. How much should you be saving each year? The following program will tell you. ADAM will ask you a series of questions regarding your school bills. It will then compute the yearly payment you should make to a savings account to accumulate the tuition funds you will need.

How to Run It: To figure out what the yearly deposits to a college savings plan should be, ADAM needs to know certain information about your child's educational plans. The computer will pose a group of questions—when your first tuition bill must be paid, how much tuition will be, and at what rate your savings will earn interest. The following guidelines will assure that the program works properly.

1) Type in dollar amounts as whole numbers without commas or dollar signs. $10,000 should be typed in as 10000, $25,000 as 25000.

2) When ADAM asks for the interest rate earned, use whole numbers with decimal places if necessary. 12.75 percent, for example, should be typed in simply as 1275.

3) The program assumes that interest on your savings compounds yearly.

FOREIGN CURRENCY CONVERTER

Purpose: What is the exchange rate between the French franc and the U.S. dollar, or between the West German mark and the U.S. dollar? Regular travelers always ask questions like these, and knowing how much your money is worth is invaluable in computing costs abroad. The program below will help you do just that. When you input the country for which you want information, ADAM automatically responds with the name of the currency for that country and the current exchange rate. You must input the exchange rates periodically in the DATA statements at the end of the program to reflect current market conditions. The DATA statements are set up to express foreign currency per U.S. dollar. Once you know the exchange rate, ADAM will ask you if you have a foreign price you'd like converted to dollars.

How to Run It: Always remember to type in the name of the country in capital letters, and don't use informal titles: Use BRITAIN for England, WEST GERMANY for Germany. Finally, don't be afraid to use decimals in foreign currency amounts; ADAM recognizes those, too.

CHECKBOOK BALANCE

Purpose: It's a chore hardly anyone likes, but balancing your checkbook is crucial. After all, it's only by reconciling the balance in your monthly bank statement with your checkbook that you can be sure your financial records accurately reflect what you have on deposit. If you're like most people, chances are you get headaches each month trying to make things match. Well, consider your checkbook balancing troubles over. Below you'll find a program that will balance your checkbook for you. By asking you a series of questions about your account, ADAM computes the sum of all charges and deposits not yet recorded by the bank and prints out a personalized worksheet balancing your account. This worksheet is similar to that found on the back of most bank statements.

How to Run It: To balance your account, ADAM needs information on deposits not yet recorded by the bank, outstanding checks, and other debits and credits that you know about but that the bank hasn't yet taken

into account, for example, accrued interest. ADAM will ask you a number of questions regarding these matters; answer them as you'd normally respond, keeping these guidelines in mind:

1) When ADAM asks for the number of checks outstanding, it doesn't mean the amount the checks were written for, but the number of checks out. For example, if you have three checks outstanding—one written for $200, a second for $250 and a third for $100—the answer to the question, "How many checks do you have outstanding?" is three. Later ADAM will ask you to input the amount of those checks.

2) A similar rule holds for unrecorded deposits—ADAM will first want to know the number of deposits not yet posted, then their amounts.

3) When ADAM asks you for information on miscellaneous charges and additions to your account not yet on your bank's books, think hard. Remember that some banks credit interest earned on a checking account only quarterly, whereas you may already have that interest recorded in your books. Moreover, if you wrote a check after the bank's closing date, the amount will appear in your checkbook register but not in your statement. Make sure that you tell ADAM everything you know about your account. It's only by having full information that ADAM can balance your monthly statement.

4) Finally, use dollars and cents but leave out the dollar sign. ADAM has been programmed to understand that, for example, 48.53 really means $48.53.

ADDRESS BOOK

Purpose: How many times have you discarded an old address book because the pages finally filled up with out-of-date addresses? How many times have you misplaced your address book? This program is designed to improve your record keeping. It provides a single place in which to store all of the addresses that often remain on scraps of paper that are soon lost. It replaces address books that are crowded with old addresses.

This program can be easily modified to keep track of many different kinds of information simply by changing some of the labels. You might want to use it to keep track of your book or record collection, or to store financial information that can be quickly and easily recalled.

How to Run It: First decide what information you're looking for. You don't necessarily have to remember the name, just one of several different parts of the address—the zip code, the city, the state, the first name or the last name.

Second, after you type RUN, keep pressing the return key until ADAM asks for the information you are prepared to give. If the name of a

city is typed after ADAM asks THE CITY IS, ADAM will print the first address found with that city. To find the next address that includes that city, type S. If you want to give ADAM another piece of information with which to begin another search, type Y.

You can put your own list of addresses into ADAM's memory by adding DATA statements after the end of the program (line 390) and before the DATA statement (line 999) that contains the word END several times. This final line tells ADAM to quit looking for more addresses. You don't have to put the addresses in any particular order, alphabetical or otherwise. If you request subsequent searches for addresses using the same piece of information, ADAM will print them beginning with the smallest line number. You do need to structure the information within an address in the following order: last name, first name, street and number, city, state, zip code. If you don't have part of the address, you still need to include something as a place holder, such as a question mark. Remember to use capital letters and not to use spaces after the commas separating the information in the address.

STATE CAPITAL TUTOR

Purpose: This is an educational program based on the flash card principle. It is designed to teach the user to remember the names of the capital cities of the United States. By changing the DATA lines, it can help someone memorize nearly any list of information—world capitals, famous persons, famous dates, athletic records, and so forth.

How to Run It: You begin by providing ADAM with some information—a success ratio, a secret number and the number of questions desired. The success ratio is a goal you set for yourself. It is a percentage and can vary from 0 to 100. The secret number is used by ADAM to generate a sequence of random numbers used to select the data for the questions. Without this, ADAM would ask the questions in the same order every time. In fact, giving ADAM the same secret number will cause the same order of questions. The last number that you need to give ADAM is the number of questions that you wish to be asked before ADAM tallies your score and compares it to your goal. Be careful to use capital letters and to check your spelling. Computers are very picky instructors.

MATHEMATICS TUTOR

Purpose: This is also an educational program, though you probably won't be able to modify it as easily as the STATE CAPITAL TUTOR. This program can provide very simple mathematical drills or very difficult ones. This is achieved by varying the number of digits in the questions. You might find this program extremely challenging.

How to Run It: As in the last program, you begin by providing ADAM with your goal and with a number that ADAM will use to generate a stream of randomly chosen numbers. You then must select the type of questions that you want to answer. You can choose addition, multiplication, subtraction or division. If you choose division, ADAM will be so impressed by your bravery that you will be asked for the whole number part only of the answer. You won't have to keep up with some of the messy fractions to which division so frequently leads. You can also decide not to continue at this point by typing in a zero. Next, you need to tell ADAM how many questions to ask and what their degree of difficulty should be. Difficulty is measured by the maximum number of digits that ADAM will use in the questions. Remember that addition and subtraction are simpler than multiplication when it comes to large numbers. Some of these questions get really tough. When you've answered all of the questions, ADAM will tell you your score and let you know if you achieved your goal. Good luck!

PROGRAM LISTINGS

Loan Amortization — Listing

```
 10 REM   PROGRAM TO COMPUTE
 15 REM   MONTHLY PAYMENTS
 20 REM   ON A CAR LOAN OR
 25 REM   HOME MORTGAGE.
 27 HOME
 30 ? "************************************************"
 40 ? "*   LOAN AMORTIZATION PROGRAM   *"
 50 ? "************************************************"
100 ?: ?
110 ? "WHAT IS THE PRICE OF THE CAR"
120 ? "OR HOUSE YOU WISH TO BUY?"
125 INPUT "   $"; a
130 ?
140 ? "HOW LARGE A DOWN PAYMENT WOULD"
150 INPUT "YOU LIKE TO MAKE $"; b
160 ?
170 ? "OVER HOW MANY YEARS WOULD YOU"
180 INPUT "LIKE TO REPAY YOUR LOAN? "; c
190 ?
200 ?
210 LET d=c*12
220 ? "WHAT ANNUAL INTEREST RATE DO"
230 INPUT "YOU EXPECT TO PAY (%) ? "; e
240 REM   COMPUTATION OF MONTHLY INTEREST RATE
250 LET f=e*.01/12
260 REM   COMPUTATION OF MONTHLY PAYMENT
270 ?
275 LET h=1
280 FOR g=1 TO d
290 LET h=h*(1+f)
300 NEXT g
310 LET i=(a-b)*f/(1-(1/h))
320 LET j=INT(100*i+.5)/100
325 ?: ?
330 ? "LOAN REPAYMENT ANALYSIS:"
```

```
335 ?: ?
340 ? "AMOUNT BORROWED IS $"; a-b
345 ?
350 ? "NUMBER OF MONTHLY PAYMENTS IS:"
355 ? "      "; d
360 ?
365 ? "MONTHLY PAYMENT YOU MUST MAKE"
366 ? "  IS $";  j
370 ?
380 ? "WOULD YOU LIKE TO MODIFY THE "
390 INPUT "TERMS OF YOUR LOAN?"; k$
400 ?: ?
410 IF k$="YES" THEN 100
420 IF k$="yes" THEN 100
425 ? "PROGRAM END"
430 END
```

Loan Amortization — Sample Output

```
*********************************************
*  LOAN AMORTIZATION PROGRAM   *
*********************************************
WHAT IS THE PRICE OF THE CAR
OR HOUSE YOU WISH TO BUY?
        $100000

HOW LARGE A DOWN PAYMENT WOULD
YOU LIKE TO MAKE $10000

OVER HOW MANY YEARS WOULD YOU
LIKE TO REPAY YOUR LOAN? 30

WHAT ANNUAL INTEREST RATE DO
YOU EXPECT TO PAY (%) ? 14

LOAN REPAYMENT ANALYSIS:

AMOUNT BORROWED IS $90000
```

```
NUMBER OF MONTHLY PAYMENTS IS:
        360

MONTHLY PAYMENT YOU MUST MAKE
  IS $1066.38

WOULD YOU LIKE TO MODIFY THE
TERMS OF YOUR LOAN?no

PROGRAM END

]

]

]

]
```

Compound Interest — Program Listing

```
 10 REM   COMPOUND INTEREST
 20 REM   PROGRAM
 22 HOME
 25 ? "*************************************************"
 26 ? "*          COMPOUND INTEREST                    *"
 27 ? "*************************************************"
 28 ?: ?
 30 ? "TO WATCH YOUR MONEY GROW OVER"
 40 ? "TIME, START BY TYPING IN THE"
 50 INPUT "INITIAL AMOUNT INVESTED $"; a
 55 ?: ?
 60 ? "NOW TYPE IN THE SIMPLE RATE OF"
 70 ? "INTEREST AT WHICH YOUR MONEY"
 80 INPUT "WILL GROW (%) "; isimple
 85 ?: ?
 90 ? "HOW MANY YEARS WILL YOU LEAVE"
100 INPUT "YOUR MONEY ON DEPOSIT?"; c
105 ?: ?
110 ? "NOW TYPE THE NUMBER OF TIMES"
```

```
120 ? "EACH YEAR THAT INTEREST WILL"
130 INPUT "COMPOUND:"; number
135 ?: ?
140 ? "HOW MANY YEARS WOULD YOU LIKE"
150 INPUT "INFORMATION ON?"; n
160 iperiod=1+((isimple/number)/100)
170 icompound=iperiod^number
175 effective =(icompound-1)*100
180 ?: ?
190 ? "THE EFFECTIVE ANNUAL RATE OF"
195 ? "INTEREST IS "; effective; " PERCENT.": ?: ?
198 ? "YR   BEGIN   INTEREST ENDING"
199 ? "      BALANCE   EARNED   BALANCE"
200 FOR k=1 TO n
240 ? k; TAB(6); "$"; INT(a+.5); TAB(15); "$";
INT((a*icompound)-a+.5); TAB(24); "$";
    INT(a*icompound+.5)
250 LET a=a*icompound
270 NEXT k
280 END
```

Compound Interest — Sample Output

```
**********************************************
*          COMPOUND INTEREST                 *
**********************************************

TO WATCH YOUR MONEY GROW OVER
TIME, START BY TYPING IN THE
INITIAL AMOUNT INVESTED $10000

NOW TYPE IN THE SIMPLE RATE OF
INTEREST AT WHICH YOUR MONEY
WILL GROW (%) 12

HOW MANY YEARS WILL YOU LEAVE
YOUR MONEY ON DEPOSIT?5
```

```
NOW TYPE THE NUMBER OF TIMES
EACH YEAR THAT INTEREST WILL
COMPOUND:12

HOW MANY YEARS WOULD YOU LIKE
INFORMATION ON?5

THE EFFECTIVE ANNUAL RATE OF
INTEREST IS 12.6825027 PERCENT.

YR      BEGIN          INTEREST       ENDING
        BALANCE        EARNED         BALANCE
1       $10000         $1268          $11268
2       $11268         $1429          $12697
3       $12697         $1610          $14308
4       $14308         $1815          $16122
5       $16122         $2045          $18167

]

]

]

]
```

College Tuition — Program Listing

```
 1 HOME
 5 ? "**************************************************"
 6 ? "*              COLLEGE TUITION                   *"
 7 ? "**************************************************"
 8 ?: ?
10 ? "HOW MUCH MONEY WOULD YOU LIKE"
15 REM  MONTHLY PAYMENTS ON
20 ? "TO HAVE STORED AWAY IN THE"
25 ? "FUTURE FOR YOUR CHILD'S HIGHER"
30 INPUT "EDUCATION? $"; a
40 ?: ?
50 ? "HOW MANY YEARS REMAIN UNTIL"
```

```
 60  ?  "YOU MUST PAY YOUR FIRST"
 70  INPUT "TUITION BILL?"; b
 80  ?: ?
 90  ?  "AT WHAT ANNUAL INTEREST RATE"
100  ?  "WILL YOUR MONEY EARN"
110  INPUT "INTEREST (%)"; c
120  LET d=c*.01
130  LET f=1
140  FOR e=1 TO b
150  LET f=f*(1+d)
160  NEXT e
170  LET g=a*d/(f-1)
180  ?: ?
190  ?  "TO MEET THOSE COLLEGE BILLS,"
200  ?  "START SAVING NOW AT AN ANNUAL"
210  ?  "RATE OF $"; INT(g); " A YEAR."
```

College Tuition — Sample Output

```
**********************************************
*              COLLEGE TUITION               *
**********************************************

HOW MUCH MONEY WOULD YOU LIKE
TO HAVE STORED AWAY IN THE
FUTURE FOR YOUR CHILD'S HIGHER
EDUCATION? $20000

HOW MANY YEARS REMAIN UNTIL
YOU MUST PAY YOUR FIRST
TUITION BILL?10

AT WHAT ANNUAL INTEREST RATE
WILL YOUR MONEY EARN
INTEREST (%)12

TO MEET THOSE COLLEGE BILLS,
START SAVING NOW AT AN ANNUAL
RATE OF $1139 A YEAR.
```

```
]
]
]
]
]
]
]
]
]
]
```

Foreign Currency Converter — Program Listing

```
  1 HOME
 10 REM   PROGRAM TO CONVERT
 20 REM   FOREIGN CURRENCIES TO
 30 REM   UNITED STATES DOLLAR AMOUNTS
 40 REM   AND TO COMPUTE DOLLAR
 50 REM   PRICES ON FOREIGN
 55 REM   GOODS
 56 ? "*************************************************"
 57 ? "*  FOREIGN CURRENCY CONVERTER   *"
 58 ? "*************************************************"
 59 ?
 60 ?: ?
 80 ? ">>>>USE CAPITAL LETTERS<<<": ?
 85 ? "ENTER THE NAME OF THE COUNTRY"
 90 ? "WHOSE CURRENCY YOU WANT"
100 INPUT "INFORMATION ON: "; a$
120 RESTORE
130 FOR b=1 TO 43
140 READ c$, d$, e
```

```
150 IF a$=c$ GOTO 260
160 NEXT b
165 ?: ?
170 ? "UNFORTUNATELY, ADAM'S DATABASE"
180 ? "HASN'T INFORMATION YET ON THE"
190 ? "CURRENCY OF "; a$; "."
200 ? "WOULD YOU LIKE TO TRY ANOTHER"
210 INPUT "COUNTRY?"; f$
220 RESTORE
230 IF f$="YES" THEN 60
240 IF f$="yes" THEN 60
250 STOP
260 ?: ?
270 ? "THE CURRENCY IN   "; c$
280 ? "IS THE "; d$; "."
290 ?: ?
300 ? "THE EXCHANGE RATE IS"; SPC(1); e
310 ? d$; "(S) PER U.S. DOLLAR."
320 ?: ?
330 ? "DO YOU HAVE A"; SPC(1); d$
340 ? "DENOMINATED PRICE YOU WOULD"
350 INPUT "LIKE CONVERTED TO U.S. DOLLARS?"; g$
360 IF g$="YES" THEN 380
370 IF g$="yes" THEN 380
372 ?: ?
373 ? "WOULD YOU LIKE TO START AGAIN?"
374 INPUT h$
375 ?: ?
376 IF h$="YES" THEN 60
378 IF h$="yes" THEN 60
379 STOP
380 ?: ?
390 ? "INPUT THE PRICE IN "; d$; "(S):"
400 INPUT i
410 ?: ?
420 ? "THE U.S. DOLLAR EQUIVALENT"
430 ? "PRICE IS $"; i/e; "."
440 RESTORE
445 GOTO 372
```

```
450 DATA      ARGENTINA, PESO, 25.502
460 DATA      AUSTRALIA, DOLLAR, 1.0826
470 DATA      AUSTRIA, SCHILLING, 19.45
480 DATA      BELGIUM, FRANC, 56.885
490 DATA      BRAZIL, CRUZEIRO, 1028.50
500 DATA      BRITAIN, POUND, .7028
510 DATA      CANADA, DOLLAR, 1.2458
520 DATA      CHINA, YUAN, 2.0686
530 DATA      COLOMBIA, PESO, 90.19
540 DATA      DENMARK, KRONE, 10.01
550 DATA      ECUADOR, SUCRE, 55.81
560 DATA      FINLAND, MARKKA, 5.86
565 DATA      FRANCE, FRANC, 8.85
570 DATA      GREECE, DRACHMA, 102.30
580 DATA      HONG KONG, DOLLAR, 7.7965
590 DATA      INDIA, RUPEE, 10.7411
600 DATA      INDONESIA, RUPIAH, 996
610 DATA      IRELAND, PUNT, .8928
620 DATA      ISRAEL, SHEKEL, 124.17
630 DATA      ITALY, LIRA, 1696
640 DATA      JAPAN, YEN, 233.60
650 DATA      LEBANON, POUND, 5.86
660 DATA      MALAYSIA, RINGGIT, 2.3360
670 DATA      MEXICO, PESO, 166
680 DATA      NETHERLANDS, GUILDER, 3.1135
690 DATA      NEW ZEALAND, DOLLAR, 1.5302
700 DATA      NORWAY, KRONE, 7.8275
710 DATA      PAKISTAN, RUPEE, 13.50
720 DATA      PERU, SOL, 2346.15
730 DATA      PHILIPPINES, PESO, 1400
740 DATA      PORTUGAL, ESCUDO, 136.0
750 DATA      SAUDI ARABIA, RIYAL, 3.51
760 DATA      SINGAPORE, DOLLAR, 2.1260
770 DATA      SOUTH AFRICA, RAND, 1.2594
780 DATA      SOUTH KOREA, WON, 799.60
790 DATA      SPAIN, PESETA, 156.45
800 DATA      SWEDEN, KRONA, 8.1160
810 DATA      SWITZERLAND, FRANC, 2.2170
820 DATA      TAIWAN, DOLLAR, 40.19
```

```
830 DATA        THAILAND, BAHT, 22.79
840 DATA        URUGUAY, NEW PESO, 46.63
850 DATA        VENEZUELA, BOLIVAR, 5.15
860 DATA        WEST GERMANY, MARK, 2.7610
870 END
```

Foreign Currency Converter — Sample Output

```
**********************************************
* FOREIGN CURRENCY CONVERTER *
**********************************************

>>>USE CAPITAL LETTERS<<<

ENTER THE NAME OF THE COUNTRY
WHOSE CURRENCY YOU WANT
INFORMATION ON: FRANCE

THE CURRENCY IN FRANCE
IS THE FRANC.

THE EXCHANGE RATE IS 8.85
FRANC(S) PER U.S. DOLLAR

DO YOU HAVE A FRANC
DENOMINATED PRICE YOU WOULD
LIKE CONVERTED TO U.S. DOLLARS?
yes

INPUT THE PRICE IN FRANC(S):
?1000

THE U.S. DOLLAR EQUIVALENT
PRICE IS $112.99435.

WOULD YOU LIKE TO START AGAIN?
?no

?Break In 379

]
```

Checkbook Balancing — Program Listing

```
  1 HOME
 10 REM   CHECKBOOK BALANCING
 20 REM   PROGRAM
 30 ? "**************************************************"
 40 ? "* CHECKBOOK BALANCING PROGRAM *"
 45 ? "**************************************************"
 46 ?: ?
 50 ? "PLEASE INPUT THE DATE OF YOUR"
 55 ? "BANK STATEMENT. ": ?
 60 ? "ENTER THE DATE WITHOUT ANY"
 70 ? "COMMAS, E.G., JANUARY 4 1984,"
 75 INPUT "THE DATE? "; a$
 80  ?: ?
 90 ? "NOW TYPE IN THE NUMBER OF"
100 ? "DEPOSITS THAT HAVEN'T BEEN"
110 ? "CREDITED TO YOUR ACCOUNT."
112 ?
115 INPUT "# OF UNCREDITED DEPOSITS? "; b
120 IF b=0 THEN 225
125 ?: ?
176 ?: ?
177 ? "INPUT THE AMOUNTS OF THOSE"
178 ? "DEPOSITS NOW:": ?
180 DIM d(b)
190 FOR c=1 TO b
200 INPUT "   $"; d(c)
210 LET e=e+d(c)
220 NEXT c
225 ?: ?
230 ? "HOW MANY CHECKS ARE THERE"
240 INPUT "OUTSTANDING? "; f
250 IF f=0 THEN 385
255 ?: ?
260 ? "NOW TYPE IN THE NUMBER OF EACH"
270 ? "OUTSTANDING CHECK, FOLLOWED BY"
280 ? "THE AMOUNT.";
285 ?: ?
```

```
315 ? "ENTER THE DATA NOW:": ?
320 j=0
330 DIM g(f)
340 DIM h(f)
350 FOR i=1 TO f
360 INPUT "   CHECK #"; g(i)
361 INPUT "   AMOUNT $"; h(i)
362 ?: ?
370 j=j+h(i)
380 NEXT i
385 ?: ?
390 ? "NOW THINK OF ANY OTHER"
400 ? "CHARGES OR ADDITIONS TO YOUR"
410 ? "ACCOUNT THAT ARE RECORDED IN"
415 ?: ?
420 ? "YOUR CHECKBOOK BUT THAT HAVE"
430 ? "NOT APPEARED IN YOUR BANK"
440 ? "STATEMENT — E.G.,CHECK"
450 ? "PRINTING FEES."
460 ?: ?
480 ? "HOW MANY ADDITIONS OR CHARGES"
490 INPUT "ARE THERE? "; k
500 IF k=0 THEN 610
510 DIM l(k)
515 ?: ?
516 ? "PLEASE TYPE IT (THEM) IN NOW:"
565 ?: ?
570 q=0
580 FOR m=1 TO k
590 INPUT "   $"; l(m)
600 q=q+l(m)
605 NEXT m
610 ?: ?
620 ? "ENTER YOUR STATEMENT BALANCE"
630 INPUT "   $"; n
635 ?
660 REM   OUTPUT OF RESULTS
770 ?: ?
776 ? "CHECKBOOK BALANCING WORKSHEET"
```

```
 777 ? "-----------------------------"
 778 ?
 780 ? TAB(16-(LEN(a$)/2)); a$
 790 ?: ?
 800 ? "CHECKS OUTSTANDING:": ?
 813 ? TAB(12); "#"; TAB(25); "AMOUNT"
 815 ?
 820 FOR o=1 TO f
 830 ? TAB(12); g(o); TAB(25); "$"; h(o)
 840 NEXT o
 850 ?
 860 ? "TOTAL"; TAB(25); "$; j
 870 ?: ?
 880 ? "DEPOSITS NOT YET RECORDED BY"
 890 ? "THE BANK:"
 900 ?
 910 FOR p=1 TO b
 920 ? TAB(25); "$" d(p)
 930 NEXT p
 940 ?: ?
 950 ? "TOTAL"; TAB(25); "$"; e
 960 ?
 965 GET t$
 970 ? "TOTAL OF UNRECORDED MISC."
 980 ? "CHARGES AND ADDITIONS:"
 990 ?
1000 ? TAB(25); "$"; q
1005 ? "BALANCE EXPECTED IN CHECKBOOK:"
1025 ?
1030 ? "   $"; n-j+e+q
1040 END
```

Checkbook Balancing — Sample Output

```
**************************************************
*  CHECKBOOK BALANCING PROGRAM   *
**************************************************

PLEASE INPUT THE DATE OF YOUR
BANK STATEMENT.

ENTER THE DATE WITHOUT ANY
COMMAS, E.G., JANUARY 4 1984,
THE DATE? FEBRUARY 7, 1984

NOW TYPE IN THE NUMBER OF
DEPOSITS THAT HAVEN'T BEEN
CREDITED TO YOUR ACCOUNT.

# OF UNCREDITED DEPOSITS? 2

INPUT THE AMOUNTS OF THOSE
DEPOSITS NOW:

     $500
     $400

HOW MANY CHECKS ARE THERE
OUTSTANDING? 2

NOW TYPE IN THE NUMBER OF EACH
OUTSTANDING CHECK, FOLLOWED BY
THE AMOUNT.

ENTER THE DATA NOW:

     CHECK #101
     AMOUNT $86

     CHECK #102
     AMOUNT $117

NOW THINK OF ANY OTHER
CHARGES OR ADDITIONS TO YOUR
ACCOUNT THAT ARE RECORDED IN

YOUR CHECKBOOK BUT THAT HAVE
NOT APPEARED IN YOUR BANK
STATEMENT — E.G., CHECK
PRINTING FEES.
```

```
HOW MANY ADDITIONS OR CHARGES
ARE THERE? 1

PLEASE TYPE IT (THEM) IN NOW:
    $25

ENTER YOUR STATEMENT BALANCE
    $1100

    FEBRUARY 7 1984

CHECKS OUTSTANDING:
            #           AMOUNT
            101         $  86
            102         $117

TOTAL                   $203

DEPOSITS NOT YET RECORDED BY
THE BANK:
                        $500
                        $400

TOTAL                   $900

TOTAL OF UNRECORDED MISC.
CHARGES AND ADDITIONS:
                        $25

BALANCE EXPECTED IN CHECKBOOK:
    $1822
```

Address Book — Program Listing

```
 10 HOME
 20 ? "*************************************************"
 30 ? "*              ADDRESS BOOK                     *"
 40 ? "*************************************************": ?: ?
 50 last$="": first$="": city$=""
 60 state$="": zip$=""
 65 ?: ? ">>> USE CAPITAL LETTERS <<<"
```

```
 70  ?: INPUT "THE LAST NAME IS "; last$
 80  IF last$<>"" GOTO 200
 90  ?: INPUT "THE FIRST NAME IS "; first$
100  IF first$<>"" GOTO 200
110  ?: INPUT "THE CITY IS "; city$
120  IF city$<>""GOTO 200
130  ?: INPUT "THE STATE IS "; state$
140  IF state$<>"" GOTO 200
150  ?: INPUT "THE ZIP CODE IS "; zip$
155  IF zip$<>"" GOTO 200
160  ?: ?: ? "TYPE 'Y' TO TRY AGAIN.": ?: ?
170  GET choice$: IF choice$<>"Y" THEN ?
     "BYE FOR NOW.": END
180  GOTO 50
200  READ a$, b$, c$, d$, e$, f$
210  IF a$=last$ GOTO 300
220  IF b$=first$ GOTO 300
230  IF d$=city$ GOTO 300
240  IF e$=state$ GOTO 300
250  IF f$=zip$ GOTO 300
260  IF a$<>"END" GOTO 200
270  ?: ?: ? "I CAN'T FIND THAT ADDRESS.": ?: ?
280  ?  TYPE 'Y' TO START AGAIN.": GET choice$
290  IF choice$="Y" THEN RUN
295  GOTO 380
300  ?: ? "---THE ADDRESS IS---"
310  ?: ? "    "; b$; "    "; a$
320  ? "    "; c$
330  ? "    "; d$; "    "; e$; "    "; f$: ?: ?
340  ? "TYPE 'S' TO SEARCH AGAIN, OR": ?
     "TYPE 'Y' TO START AGAIN."
350  GET choice$
360  IF choice$="Y" THEN RUN
370  IF choice$="S" GOTO 200
380  ?: ?: ? "GOODBYE. I LOOK FORWARD TO": ?
     "OUR NEXT MEETING."
390  END
600  DATA      SMITH,ALAN,620
               HEMLOCK,BARSTOW,TX,79797
```

```
601  DATA     SMITH,JUDY,12 KODALY PLACE,PALO
              ALTO,CA,94949
602  DATA     MEEK,BEN,606 SHYLOCK
              ST.,LONDON,ENGLAND, !!!!!
603  DATA     HORN,STEVE,15 LEHMAN LANE,NEW
              YORK,NY,10011
604  DATA     HAMMER,M. & K., 76 TRIPLET
              CIRCLE,NOME,ALASKA,00001
605  DATA     GROSS,BRIAN,1600 PENNSYLVANIA
              AVE.,WASHINGTON,DC,23232
999  DATA     END,END,END,END,END,END
```

Address Book — Sample Output

```
**********************************************
*              ADDRESS BOOK                  *
**********************************************

>>> USE CAPITAL LETTERS <<<

THE LAST NAME IS

THE FIRST NAME IS

THE CITY IS BARSTOW

---THE ADDRESS IS---

   ALAN SMITH
   620 HEMLOCK
   BARSTOW TX 79797

TYPE 'S' TO SEARCH AGAIN, OR
TYPE 'Y' TO START AGAIN.

**********************************************
*              ADDRESS BOOK                  *
**********************************************

>>> USE CAPITAL LETTERS <<<
```

```
THE LAST NAME IS HAMMER

---THE ADDRESS IS---

   M. & K. HAMMER
   76 TRIPLET CIRCLE
   NOME ALASKA 00001

TYPE 'S' TO SEARCH AGAIN, OR
TYPE 'Y' TO START AGAIN.
```

State Capital — Program Listing

```
 10 DIM state$(50): DIM capital$(50)
 20 FOR i=1 TO 50
 30 READ state$(i), capital$(i)
 35 NEXT i
 40 HOME
 50 ? "************************************************"
 60 ? "*          STATE CAPITAL TUTOR          *"
 70 ? "************************************************": ?: ?
 80 right=0
 90 ? "WHAT SUCCESS RATIO (PERCENT)": INPUT "DO
    YOU HOPE TO ACHIEVE? "; goal: ?
100  ? "TELL ME A SECRET NUMBER FOR MY": INPUT
    "RANDOM NUMBER GENERATOR. "; secret: ?
110 IF secret>1000 THEN  ? "THAT'S TOO BIG FOR ME
    TO KEEP": ? "SECRET.": ?: GOTO 100
120 FOR m=1 TO INT(secret)
130 a=RND(m)
140 NEXT m
150 ? "HOW MANY QUESTIONS DO YOU": INPUT "WANT
    ME TO ASK? "; number: ?
155 ? ">>> USE UPPER CASE <<<": ?
160 FOR j=1 TO number
170 x=INT(.5+(49*RND(secret)))+1
180 secret=secret+number
```

```
190  ?  "FOR THE STATE OF "; state$(x); ","
200  INPUT "THE CAPITAL IS "; answer$: ?
210  IF answer$=capital$(x) GOTO 240
220  ?  "THAT'S NOT QUITE RIGHT.   THE": ? "CAPITAL
     OF "; state$(x): ? "IS "; capital$(x); ".": ?: ?
230  GOTO 260
240  ?  "THAT'S RIGHT!": ?: ?
250  right=right+1
260  NEXT j
270  percent=INT(100*(right/number)+.5)
280  IF percent<goal GOTO 320
290  ?  "*******************************": ?: ?
    "CONGRATULATIONS! YOU HAVE"; ? "EXCEEDED
    YOUR GOAL OF "; goal
300  ?  "PERCENT WITH A SCORE OF "; percent; ".": ?: ?
     "*******************************": ?
310  GOTO 340
320  ?  "@@@@@@@@@@@@@@@@@@@@@@@@@
     @@@@@@@@": ?: ? "BETTER LUCK NEXT
     TIME.   YOU": ? "CORRECTLY ANSWERED ONLY ";
     right
325  ?  "OF "; number; " QUESTIONS ASKED.": ?
330  ?  "THAT GIVES YOU A SCORE OF "; percent; ",": ?: ?
     "@@@@@@@@@@@@@@@@@@@@@@@@@@@@
     @@@@@": ?
340  ?  "TYPE THE LETTER 'Y' TO TRY": ? "AGAIN."
350  GET choice$: IF choice$="Y" GOTO 40
360 ?: ?: ? "HURRY BACK. I'LL BE WAITING."
370  END
401  DATA     ALABAMA,MONTGOMERY
402  DATA     ALASKA,JUNEAU
403  DATA     ARIZONA,PHOENIX
404  DATA     ARKANSAS,LITTLE ROCK
405  DATA     CALIFORNIA,SACRAMENTO
406  DATA     COLORADO,DENVER
407  DATA     CONNECTICUT,HARTFORD
408  DATA     DELAWARE,DOVER
409  DATA     FLORIDA,TALLAHASSEE
410  DATA     GEORGIA,ATLANTA
```

```
411 DATA    HAWAII,HONOLULU
412 DATA    IDAHO,BOISE
413 DATA    ILLINOIS,SPRINGFIELD
414 DATA    INDIANA,INDIANAPOLIS
415 DATA    IOWA,DES MOINES
416 DATA    KANSAS,TOPEKA
417 DATA    KENTUCKY,FRANKFORT
418 DATA    LOUISIANA,BATON ROUGE
419 DATA    MAINE,AUGUSTA
420 DATA    MARYLAND,ANNAPOLIS
421 DATA    MASSACHUSETTS,BOSTON
422 DATA    MICHIGAN,LANSING
423 DATA    MINNESOTA,ST. PAUL
424 DATA    MISSISSIPPI,JACKSON
425 DATA    MISSOURI,JEFFERSON CITY
426 DATA    MONTANA,HELENA
427 DATA    NEBRASKA,LINCOLN
428 DATA    NEVADA,CARSON CITY
429 DATA    NEW HAMPSHIRE,CONCORD
430 DATA    NEW JERSEY,TRENTON
431 DATA    MEW MEXICO,SANTA FE
432 DATA    NEW YORK,ALBANY
433 DATA    NORTH CAROLINA,RALEIGH
434 DATA    NORTH DAKOTA,BISMARCK
435 DATA    OHIO,COLUMBUS
436 DATA    OKLAHOMA,OKLAHOMA CITY
437 DATA    OREGON,SALEM
438 DATA    PENNSYLVANIA,HARRISBURG
439 DATA    RHODE ISLAND,PROVIDENCE
440 DATA    SOUTH CAROLINA,COLUMBIA
441 DATA    SOUTH DAKOTA,PIERRE
442 DATA    TENNESSEE,NASHVILLE
443 DATA    TEXAS,AUSTIN
444 DATA    UTAH,SALT LAKE CITY
445 DATA    VERMONT,MONTPELIER
446 DATA    VIRGINIA,RICHMOND
447 DATA    WASHINGTON,OLYMPIA
448 DATA    WEST VIRGINIA,CHARLESTON
449 DATA    WISCONSIN,MADISON
450 DATA    WYOMING,CHEYENNE
```

State Capital Tutor — Sample Output

```
**************************************************
*          STATE CAPITAL TUTOR                   *
**************************************************

WHAT SUCCESS RATIO (PERCENT)
DO YOU HOPE TO ACHIEVE? 33

TELL ME A SECRET NUMBER FOR MY
RANDOM NUMBER GENERATOR. 345

HOW MANY QUESTIONS DO YOU
WANT ME TO ASK? 3

>>> USE UPPER CASE <<<

FOR THE STATE OF FLORIDA,
THE CAPITAL IS TALLAHASSEE

THA'S RIGHT!

FOR THE STATE OF NORTH DAKOTA,
THE CAPITAL IS BISMARCK

THAT'S RIGHT!

FOR THE STATE OF ARIZONA,
THE CAPITAL IS TIMBUKTU

THAT'S NOT QUITE RIGHT. THE
CAPITAL OF ARIZONA
IS PHOENIX.

**************************************************

CONTRATULATIONS! YOU HAVE
EXCEEDED YOUR GOAL OF 33
PERCENT WITH A SCORE OF 67.
**************************************************

TYPE THE LETTER 'Y' TO TRY
AGAIN.
```

Mathematics Tutor — Program Listing

```
 5 HOME
10 REM   MATHEMATICS TUTOR
15 REM   ---------------------------
20 REM   This program will test
30 REM your mathematical
40 REM   skills and record
50 REM   your success ratio
51 REM   ---------------------------
52 REM   ---------------------------
60 ? "************************************************"
70 ? "*          MATHEMATICS TUTOR          *": ?
     "************************************************"
80 right=0
90 ?: ? "WHAT SUCCESS RATIO DO YOU HOPE": INPUT
   "TO ACHIEVE? "; goal
100 ?
110 ? "TELL ME A SECRET NUMBER FOR MY": INPUT
   "RANDOM NUMBER GENERATOR. "; secret: ?
120 IF secret>1000 THEN ? "THAT'S TOO BIG FOR ME TO
   KEEP": ? "SECRET.": ?: GOTO 100
210 ?
220 ? "      (1) ADDITION"
230 ? "      (2) SUBTRACTION"
240 ? "      (3) DIVISION"
250 ? "      (4) MULTIPLICATION"
260 ?
270 ?
280 ? "PICK THE CORRESPONDING NUMBER"
290 ? "TO CHOOSE THE TYPE OF PROBLEM"
300 INPUT "OR INPUT 0 TO END. "; type: ?
305 IF type=0 THEN ? "SEE YOU LATER,
   ALLIGATOR.": END
310 INPUT "HOW MANY PROBLEMS? "; number: ?
320 ? "WHAT IS THE MAXIMUM NUMBER OF"
330 INPUT "DIGITS IN EACH NUMBER? "; digits: ?
331 IF digits<6 GOTO 334
332 ? "YOU'RE OUT OF MY LEAGUE. TRY"
```

```
333 ? "A SMALLER NUMBER.": ?: GOTO 320
334 IF digits=1 THEN  factor=10
335 IF digits=2 THEN  factor=100
336 IF digits=3 THEN  factor=1000
337 IF digits=4 THEN  factor=10000
338 IF digits=5 THEN  factor=100000
339 IF digits=6 THEN  factor=1000000
340 FOR m=1 TO INT(secret)
341 a=RND(m)
342 NEXT m
350 IF type=1 GOTO 400
360 IF type=2 GOTO 600
370 IF type=3 GOTO 800
380 IF type=4 GOTO 1000
390 ?: ? "I DON'T UNDERSTAND WHAT TYPE": ? "OF
    PROBLEM YOU WANTED. PLEASE": ? "TRY
    AGAIN.": GOTO 210
400 REM  THIS IS THE ADDITION
410 REM  SECTION
420 FOR i=1 TO number
430 a=RND(secret*number)
440 b=INT(a*factor)
450 c=INT((a*factor-b)*factor)
460 d=b+c
470 ? "WHAT IS "; b; " PLUS "; c
480 INPUT "YOUR ANSWER IS: "; answer: ?
490 IF answer=d GOTO 530
500 ? "THAT ISN'T QUITE RIGHT."
510 ? "THE ANSWER IS ACTUALLY "; d: ?
520 GOTO 550
530 right=right+1
540 ? "THAT'S RIGHT!!": ?
550 NEXT i
560 GOTO 1160
600 REM  THIS IS THE SUBTRACTION
610 REM  SECTION
620 FOR i=1 TO number
630 a=RND(secret*number)
640 b=INT(a*factor)
```

```
650 c=INT((a*factor-b)*factor)
660 d=b-c
670 ? "WHAT IS "; b; " MINUS "; c
680 INPUT "YOUR ANSWER IS: "; answer: ?
690 IF answer=d GOTO 730
700 ? "THAT ISN'T QUITE RIGHT."
710 ? "THE ANSWER IS ACTUALLY "; d: ?
720 GOTO 750
730 right=right+1
740 ? "THAT'S RIGHT!!": ?
750 NEXT i
760 GOTO 1160
800 REM   THIS IS THE DIVISION
810 REM   SECTION
820 FOR i=1 TO number
830 a=RND(secret*number)
840 b=INT(a*factor)
850 c=INT((a*factor-b)*factor)
860 d=INT(b/c+.5)
865 ?: ? "GIVE YOUR ANSWER TO THE": ? "NEAREST
    WHOLE NUMBER. ": ?
870 ? "WHAT IS "; b; " DIVIDED BY "; c
880 INPUT "YOUR ANSWER IS: "; answer: ?
890 IF answer=d GOTO 930
900 ? "THAT ISN'T QUITE RIGHT."
910 ? "THE ANSWER IS ACTUALLY "; d: ?
920 GOTO 950
930 right=right+1
940 ? "THAT'S RIGHT!!": ?
950 NEXT i
960 GOTO 1160
1000 REM   THIS IS THE MULTIPLICATION
1010 REM   SECTION
1020 FOR i=1 TO number
1030 a=RND(secret*number)
1040 b=INT(a*factor)
1050 c=INT((a*factor-b)*factor)
1060 d=b*c
1070 ? "WHAT IS "; b; " MULTIPLIED BY "; c
```

```
1080 INPUT "YOUR ANSWER IS: "; answer: ?
1090 IF answer=d GOTO 1130
1100 ? "THAT ISN'T QUITE RIGHT."
1110 ? "THE ANSWER IS ACTUALLY "; d: ?
1120 GOTO 1150
1130 right=right+1
1140 ? "THAT'S RIGHT!!": ?
1150 NEXT i
1160 percent=INT(right/number*100+.5)
1170 ?
1180 IF percent<goal GOTO 1240
1190 ? "CONGRATULATIONS!!": ?
1200 ? "YOU ACHIEVED YOUR GOAL OF "; goal: ?
     "PERCENT.": ?
1210 ? "YOU ANSWERED "; right; " OF "; number; "
     QUESTIONS": ? "ASKED. THAT'S A SUCCESS RATIO"
1220 ? "OF "; percent; " PERCENT."
1230 GOTO 1270
1240 ? "UNFORTUNATELY YOU DIDN'T": ? "ACHIEVE
     YOUR GOAL. YOU ONLY"
1250 ? "ANSWERED "; right; " OF THE "; number: ?
     "QUESTIONS I ASKED, FOR A "; percent
1260 ? "PERCENT SUCCESS RATIO."
1270 ?: ? "TYPE THE LETTER 'y' TO TRY": ? "AGAIN."
1280 GET choice$: IF choice$="y" GOTO 80
1290 ?: ? "GOODBYE. I HOPE TO PLAY WITH": ? "YOU
     AGAIN VERY SOON.": END
```

Mathematics Tutor — Sample Output

```
**********************************************
*          MATHEMATICS TUTOR                 *
**********************************************

WHAT SUCCESS RATIO DO YOU HOPE
TO ACHIEVE? 50

TELL ME A SECRET NUMBER FOR MY
RANDOM NUMBER GENERATOR. 555

     (1)  ADDITION
     (2)  SUBTRACTION
     (3)  DIVISION
     (4)  MULTIPLICATION

PICK THE CORRESPONDING NUMBER
TO CHOOSE THE TYPE OF PROBLEM
OR INPUT 0 TO END. 1

HOW MANY PROBLEMS? 4

WHAT IS THE MAXIMUM NUMBER OF
DIGITS IN EACH NUMBER? 2

WHAT IS 85 PLUS 30
YOUR ANSWER IS: 115

THAT'S RIGHT!!

WHAT IS 46 PLUS 30
YOUR ANSWER IS: 76

THAT'S RIGHT!!

WHAT IS 79 PLUS 97
YOUR ANSWER IS: 100

THAT ISN'T QUITE RIGHT.
THE ANSWER IS ACTUALLY 176
```

WHAT IS 3 PLUS 49
YOUR ANSWER IS: 52

THAT'S RIGHT!!

CONGRATULATIONS!!

YOU ACHIEVED YOUR GOAL OF 50 PERCENT.

YOU ANSWERED 3 OF 4 QUESTIONS ASKED. THAT'S A SUCCESS RATIO OF 75 PERCENT.

TYPE THE LETTER 'y' TO TRY AGAIN.

GOODBYE. I HOPE TO PLAY WITH YOU AGAIN VERY SOON.

]

]

CHAPTER 11
MAINTENANCE AND TROUBLE-SHOOTING

Care and Maintenance of ADAM

You will be glad to know that ADAM is an unusually hearty computer. Unlike the computing machines of yesteryear, which used oversized vacuum tubes that often burned out after extended use, ADAM is made almost entirely of solid-state circuitry—resistors, transistors, diodes and other tiny electronic parts that remain cool and effective for years. Strong plastic has replaced metals subject to corrosion and the gigantic core memories that computers used to use have given way to small, easy to use, sturdy tape memory systems like ADAM's. For all of these reasons, maintaining ADAM is easy. Your home computer should give you years of service and enjoyment if you follow a few guidelines:

1. Keep your television screen or monitor free of dust and lint. To clean the screen, use a commercial glass cleaner like Windex or combine one quart of warm water with two or three tablespoons of ammonia. Apply the mixture lightly to the screen, being careful to dry the surface afterward.

2. To clean ADAM's keyboard and memory console, dust them occasionally with a soft, dry cloth. For safety reasons it's important that you keep all electrical wires dry and in good working condition, and that you keep all jewelry, hair and other objects away from the inside of the printer. It goes without saying that if you're going to smoke or drink near your computer—habits we don't recommend—be sure to keep whatever you're imbibing away from the machine's interior.

3. ADAM's tapes are extremely sensitive. Since information is stored on the tapes through a magnetic process, it's important that you keep the tapes away from any magnet—otherwise what you save may become garbled. This means keeping the tapes away from the screen or printer; both generate electromagnetic fields that can cause gobbledygook on tapes. This also means that you shouldn't leave any tapes in the tape drives when you're switching the machine on and off. The surge of current accompanying the switching also creates a magnetic field that can convert a prize-winning novel into nonsense. Always remove the tapes before you switch the power on or off, and always store the digital data packs in a safe, dry place, away from extreme heat or cold.

Trouble-shooting

Sometimes, of course, even machines that have been well maintained have their problems. In cases of malfunction in which you can't make the repairs, you'll have to ship ADAM to either Coleco's West Hartford, Connecticut headquarters or, most likely, to a computer repair center of the Honeywell Corporation. Coleco has entered into a deal with Honeywell under which Honeywell will service ADAM machines under warranty. To find out about the center nearest you, call Coleco's toll-free number, (800) 842-1225, from 8 a.m. to 5 p.m. You can also call another number, (203) 521-7222, during the same hours. This number, though not toll-free, connects directly to Coleco's technicians, who may be able to help you solve the problem yourself. They are usually very knowledgeable and helpful. Another advantage is that this number is not usually as busy as the toll-free number.

For the sake of convenience, you'll probably want to try to do some repair work yourself, particularly if the problem is minor. Many problems can be resolved by consulting the following trouble-shooting chart:

Symptom	Remedy
Left digit of numbers and right letter of words are missing.	Adjust horizontal control on television set or monitor.
Screen filled with "snow," not SmartWriter opening menu.	Check to make sure television is set to channel 3 or 4. Make sure switch box on back of television is set to computer.
Printed text is garbled.	Check to make sure ribbon and daisy wheel typing element have been correctly installed.
SmartBASIC program won't store after changes are made.	Program has been locked into ADAM's memory using the LOCK command. Use the UNLOCK command before trying to save a program.
Programs won't store because digital data pack is full.	Free up space on the tape by deleting old copies of programs.

The ribbon in the printing cartridge is jammed, causing uneven printing.	Take the cartridge out of the printer. Carefully separate the top half of the cartridge from the bottom half. You may have to use a flat-headed screwdriver, but be careful not to break the plastic. Unjam and rewrap the ribbon. From above and with the printing area away from you, the ribbon moves from right to left like a cassette tape. You can discard the ribbon on the left spool and resplice the remaining end onto the left spool. Now put the two halves back together. This is a messy job.
The ribbon continues to jam or the printing remains uneven.	The printing is uneven when the ribbon isn't pulled through easily. The uneven pulling also causes the jamming. One of the causes can be the little black rubber band around the two white plastic knobs on the top of the cartridge. This band turns the closer knob to wind the ribbon on the left spool as ADAM turns the farther knob from beneath the cartridge to pull the ribbon from the other spool. As more and more ribbon is used, the rate at which the closer knob can turn changes. The black band can be so tight that the turning of the farther knob is restricted by the slower turning closer knob. One possible solution is to use another rubber band that is tight enough to pull on the closer knob yet doesn't restrict the farther knob. The band

	we use is a plain brown rubber band costing less than a penny and measuring about ¾" in diameter. It is smaller than the black band, which is about 1¼" across, but is more elastic and therefore doesn't provide as much friction. This band will wear out more quickly, but is a good value for a penny.
Printer overheats, or whines.	Turn ADAM off for one half hour for every two hours of use. If the printer continues to whine when the machine is turned on again, rotate paper roller on printer, advancing paper until whine stops.

GLOSSARY

Like any field, computer science has its shop talk. The computer world is filled with technical terms, abstract concepts and obscure phrases that make learning difficult even for the advanced user. In many respects, the vocabulary of computers might as well be Greek—it's just as much a foreign tongue. Even the simplest concept can be made difficult when couched in complex terms.

But however arduous a task it may be to learn these terms, we think that getting to know the vocabulary of computers is worthwhile. The reason: Only by understanding some of the technical terms of computers can you begin to comprehend how computers work. This is as much true for small, inexpensive home computers like ADAM as it is for big computers used in large corporations. Even if you don't plan to use these words in everyday speech, chances are that someone you're talking to will. This makes it all the more important for you to learn the lingo of computers, if only to make you an intelligent conversationalist in computer circles.

What follows, then, is a glossary of computer terms. This dictionary should provide you with a working definition of the most commonly used words and phrases in computer science. Many of the words should be familiar to you since they refer to parts and aspects of ADAM that we've discussed throughout the book. In addition to these words, we've tried to provide you with a list of other computer terms not specifically connected to ADAM that will enrich your overall computer vocabulary and make you a more intelligent computer user, whether it be of ADAM or another machine. Whenever possible, we've attempted to provide plain English, easy-to-understand definitions that use analogies to make the point.

A

ACCESS The ability to summon a file from a computer's memory or to make changes in it. Whenever you call a file onto your television screen, you are said to be accessing that file.

ACCESS TIME This is the number of seconds required to access a file from memory. With ADAM, access time is unusually quick because files are stored on a high-speed tape. Other, costlier computers use disk drives that enjoy even shorter access times.

ACOUSTIC COUPLER Sometimes known simply as a coupler, this is a device that enables computers like ADAM to transmit information over telephone lines. Typically, an acoustic coupler has two plastic cups

designed to hold the two ends of a telephone receiver. When the coupler is connected to a telephone and a computer, information flows from the computer to the coupler and then over the telephone lines. See also *modem*.

ALPHANUMERIC EXPRESSION A term or phrase containing letters and numbers. An example of such an expression would be an address.

APPLICATION SOFTWARE Computer programs that permit you to apply your computer's abilities to specific uses, for example, a word processor such as ADAM's SmartWriter program, which allows you to write a letter, or a personal finance program, which helps you to balance your checkbook.

ARITHMETIC EXPRESSION A group of numbers and/or numeric variables that when used in conjunction with an arithmetic operator such as addition or subtraction tell the computer to perform a specific mathematical task. For example:

```
2+2
5-3
2*2
3/7
A+B
```

are all arithmetic expressions. In the last example, it is assumed that the user has defined the variables A and B.

ARITHMETIC LOGIC UNIT Commonly abbreviated as ALU, the arithmetic logic unit is the part of a computer's electronic brain where arithmetic operations such as multiplication and division are performed. The ALU is considered part of a computer's *central processing unit* (which see also)—the electronic brain itself.

ARITHMETIC OPERATION Any of the following four mathematical procedures: addition, subtraction, multiplication and division. Even low cost home computers like ADAM can perform all of these operations using large numbers.

ARTIFICIAL INTELLIGENCE Also known simply as AI, this term refers to the increasing ability of computers to understand normal English speech, to learn, and to make decisions. Since computers are machines made by man, they will probably never be able to think in the human sense. However, scientists are increasingly able to teach computers to remember experiences so that they do not repeat mistakes.

ASCII An acronym for American Standard Code for Information Interchange, ASCII is a system for converting ordinary English text, including BASIC programming language commands, into something the computer can understand. Based on the binary (base 2) method of counting, the ASCII system translates each letter, number or symbol into a binary digit stored in the computer's memory. Then, when the computer wants to display something, it converts back from binary to regular speech using the ASCII system. Virtually all modern computers depend on ASCII for operations.

B

BACKUP Typically used in reference to files in a computer's memory, a backup is an extra electronic copy of a program, text or set of instructions that you have given the computer. You make a backup file just in case you accidentally erase an original file from ADAM's memory. Since power blackouts and brownouts, magnetic fields and other uncontrollable forces and events can erase memory, it's critical to make backups of your most important computer files.

BASE 2 Also known as the *binary* method of counting, base 2 is the counting system that computers use. Base 2 has only two digits, 0 and 1, yet it is possible to express all numbers using these digits. For example, three is written 11 (one times two, plus one), the number five is written 101 (one times two squared, plus zero times two, plus one). Base 2 facilitates computer arithmetic, since the electronic switches inside a computer can be either open or closed, corresponding to zero or one in binary.

BASE 10 This is the numbering system we use in everyday life. The base 10 system consists of 10 digits, 0 through 9. It's important to note that although base 10 is the most commonly used counting system, it is by no means the only one. Other counting systems, such as *base 2* (see) and base 16, are used in numerous scientific and technical applications.

BASIC An acronym for Beginner's All-purpose Symbolic Instruction Code, BASIC is a programming language that is rapidly emerging as the most commonly used computer language. Consisting of numerous computer commands, some common English phrases and a grammar, BASIC enables the user to communicate with his computer. ADAM uses a version of BASIC known as SmartBASIC. With SmartBASIC, programming errors are pointed out as a program is being written, not at the end of a work session.

BAUD A measure of how quickly information is being transmitted, the term is typically used in reference to sending data over a telephone line from one computer to another. For microcomputers, a baud is approximately one bit per second, so a communications device transmitting at 300 baud is sending data at the rate of about 300 bits per second. This translates into a transmission rate of approximately 5 words a second or 300 words a minute.

BIT Short for binary digit, a bit is the smallest unit of computer memory. It has a value of 1 or 0. (See Base 2.) To store a number inside its memory, ADAM uses 8 bits, even if in most cases the bits are 0. For example:

00000011

is an 8-bit binary expression for the number 3. This is how ADAM would store the number 3 in its memory.

BOARD Short for printed circuit board. A board contains the transistors, resistors, capacitors and other electronic components that together make up the internal mechanisms of a computer. More precisely, electronic components are soldered onto a board and linked by ultrathin strips of metal printed on the board. A board is typically made of a material that does not conduct electricity, such as plastic.

BOOT This is a verb that basically means to load the program that is a computer's operating system into the computer's working memory. In other words, by booting up you give the computer the initial instructions necessary to obey other commands or to carry out other computer programs. It's possible with some computers, including ADAM, to have a program booted automatically when the machine is turned on. This term comes from the word "bootstrap" as in "pulling oneself up by one's own bootstraps," because of the self-starting nature of the operation.

BUFFER This is a place in a computer's memory or in a printer where a program, data or text can be stored temporarily while the computer carries out another task. Buffers are useful because they enable the computer user to start on another computing task while the computer is still working on the last task. For example, it's possible to order a computer to repint a text file, and then to begin writing another piece of text while the first piece is being printed. The contents of the first text are being stored in a buffer.

BUG A bug in computerese is not an insect. Rather, a bug is an error in a computer program that causes that program either to work incorrectly or not to work at all. Advanced computer users speak of debugging their programs, that is, getting the bugs out.

BYTE A byte is nothing more than 8 bits, or binary digits. Typically, computers like ADAM use a full 8 bits—a complete byte—to store one character, letter, or symbol in memory. The word ADAM, for example, would consume 4 bytes of memory, one for each letter: A, D, A and M.

C

CANNED SOFTWARE A technical term for a computer program that has already been written and is packaged and sold in computer stores with instructions as a single unit. Typically, canned software consists of applications programs that serve a specific function. for example, text editing.

CARTRIDGE Just as the word implies, a cartridge is a container, generally plastic, that holds a tape that computers can read. Cartridges are similar to cassettes with the one exception that cartridges generally can only be read and not used as storage devices to record programs. Most of the games you'll play with ADAM come on cartridges.

CATALOG A catalog (or directory) is a list of the names of programs and other files currently on storage in a computer's memory. A computer user asks for a directory to see what's in storage and, equally important, to see how much storage lies vacant for additional use.

CATHODE RAY TUBE Typically called a CRT, a cathode ray tube is the main part of a television set or other monitor that forms the picture on the screen. Among computer users, a cathode ray tube has emerged as the name given to any monitor that uses a CRT to display an image, whether it be text or pictures. If you don't want to remember all these technical aspects, you can simply think of the CRT as your television set.

CENTRAL PROCESSING UNIT Also known simply as the CPU, the central processing unit is the electronic brain of computers like ADAM. It is in the CPU that instructions, in the form of a computer program, are carried out and numbers are manipulated. The CPU also contains a computer's operating system—the internal program that coordinates activities among the computer's various parts. The CPU typically takes the form of a tiny chip of silicon, and is also called a microprocessor. All of the circuitry that goes into the CPU lies on this silicon chip.

CHIP Short for silicon chip, a chip is a surface, typically made of silicon, onto which electric circuits are etched. One particularly complicated chip is ADAM's electronic brain, known as the CPU or microprocessor. (See also *central processing unit.*) This particular chip has literally hundreds of tiny electronic components soldered onto it that together compose various electrical circuits.

COMMAND This is simply an instruction that orders the computer to perform a task. Typically a command will consist of several parts. For example, a multiplication command may tell ADAM to multiply two numbers, then tell ADAM where to store the result. An example of such a command in SmartBASIC would be:

```
10 X=2*3
```

On the other hand, some commands, such as LIST, just have one part to them. The LIST command tells ADAM to list out all the parts of a program you have written.

COMPILER An internal computer program that translates computer languages such as BASIC into machine language that computers can understand.

COMPUTER GRAPHICS Graphs, charts, product designs, pictures and any other displays that use a computer's ability to draw and provide color images on a television set. ADAM has the ability to provide some limited graphics.

COMPUTER PROGRAM A logical series of instructions, in a computer language, that tell the computer what to do.The computer carries out each line of a program in order, and each program generally has a specific purpose. For example, the following program, written in SmartBASIC, is designed to get ADAM to display the words to a favorite American song.

```
10 Print "Row, row, row your bo
at, gently down the stream."

20 Print "Merrily, merrily, me
rrily, merrily, life is but a d
ream."

30 End
```

COMPUTER SYSTEM A computer system refers to a computer used in conjunction with external devices, such as a printer or screen. Each part—the computer, the printer and the screen—is considered a component. Together, the components make up the computer system. More advanced computer systems include such components as disk drives for memory and plotters to draw graphs and charts. However costly and complex, a computer system always processes data and other information and then communicates it to the outside world.

CPU See *central processing unit.*

CRT See *cathode ray tube.*

CURSOR This is a tiny flash of light that tells you where you are currently working on your screen. In the case of word processing programs, such as ADAM's SmartWriter, the cursor tells you where the next character will appear on the screen when it is typed. Similarly, the cursor also indicates where the next character will appear when typing a program in a programming language, such as SmartBASIC. Each time a key is hit, the cursor advances one position on the screen. ADAM's cursor takes the form of a brief underline.

D

DAISY WHEEL PRINTER This is a printer that relies on a special type of printing element known as a daisy wheel to create images on a regular page. The element is called a daisy wheel because it is round and has long thin strips extending out from its core, with the strips arranged like the petals of a daisy. Each strip has a number or letter on its end. When a key is struck, the daisy wheel spins to the point of the requested character, locks in position and strikes against a ribbon, thus leaving an image on the page. ADAM comes with a letter-quality, daisy wheel printer.

DATA Any set of numbers, letters or words that serves as a source of raw information to be manipulated in the course of executing a computer program. For example, if you write a program to add two numbers, you might list various candidates at the end of your program. Such numbers would form your data.

DATA PROCESSING This is the formal name given to the calculations and other data manipulations that take place inside computers like ADAM. Indeed, all that computers like ADAM do is take data and convert it into a more usable form.

DEBUG To remove the *bugs* (which see also)—the misspellings, errors of logic, and omissions that prevent the program from performing its assigned task—from a computer program. Debugging can be a painstaking process. It consists of sifting through a program line by line in search of errors.

DIRECTORY See *catalog.*

DISKETTE Also called a disk. A square flexible sheet about 5¼ inches across diagonally that has become the most common medium on which computers store information. You can think of disks as being similar to blank cassette tapes in the sense that information is recorded on them and played back from them.

DISPLAY Any output coming from a computer and presented on a monitor or television set. An example of a display would be a letter that appears on your screen. (See also *cathode ray tube*.)

DOT MATRIX PRINTER Unlike ADAM's daisy wheel printer, a dot matrix printer forms images out of tiny dots grouped together to make letters or numbers. Some dot matrix printers use a print head that strikes against a ribbon to form dots on a page; others literally spray the dots onto the page in the form of wet ink. Dot matrix printers typically work many times faster than letter-quality printers like ADAM's; quality is sacrificed for speed and cost.

DOWN TIME A splendid euphemism for the time when computers break down or are being modified and therefore are unavailable for use.

E

EDIT A broadly defined computer term meaning to change the contents of a file. It can mean (in the case of word processing) to correct the spelling, grammar or punctuation of a letter or other text. It can also mean adding lines, moving lines, or making any other revisions to a text or program. Indeed, whenever you summon a file to your screen and make changes you are editing.

ERASE To eliminate a file from memory. You can think of erasing from a computer's memory as being analogous to erasing words on a blackboard; once the words have been deleted they are gone forever, not to be easily replaced. This makes it important for you to think carefully before you erase material from ADAM's memory.

EXECUTE To instruct a computer to carry out a program or other instructions. More specifically, you use the RUN command. When ADAM encounters the word RUN, it immediately begins reading through the lines of a program and carries out the instructions in sequence.

F

FILE A file is the name given to an electronic storage space in a computer's memory. When you prepare a program or piece of text and then store it in memory, the program is said to be stored in a file. Each file

has a name that you assign. In fact, the way to think of files is to think of a filing cabinet with information stored inside it. This filing cabinet is analogous to the computer's memory, and the file folders that go inside the cabinet correspond to electronic files in the computer's memory.

FLOPPY DISK See *diskette.*

FORMATTING The process of electronically organizing a blank disk into sectors and regions so that information can be stored on the disk in an ordered manner that makes retrieval easy.

FUNCTION KEY A key on a computer keyboard that when pressed tells the computer to perform a specific task. Generally, function keys can be preprogrammed so that they do the same job each time they are pressed. In the case of ADAM, the Smart Keys labeled with Roman numerals are function keys.

G

GRAPHICS Any information displayed by a computer on a screen that is not text or programming language. More precisely, graphics are charts, graphs and other displays that form a picture on your screen to convey a message. The message can be a mathematical one—for example, a pie chart showing market share of various manufacturers in the home computer field—or graphics can take the form of animated figures formed from the computer's ability to project and combine colored dots on a screen. ADAM has the capacity to produce limited graphics.

H

HARD COPY A copy of a text, a program, data or other matter from ADAM's memory that has been printed out and is now in permanent form that can be held in one's hand. A hard copy can be contrasted to an electronic copy on ADAM's screen or in ADAM's memory. Unlike hard copies, electronic copies can be easily and neatly edited.

HARDWARE This is the term used to refer to all the physical components of a computer system, including the computer itself, and the printer, monitor, keyboard and other attachments. Software by comparison, is the technical term for computer programs, the instructions that tell the computer what to do.

HIGH-LEVEL LANGUAGE A computer language that relies on familiar English words to convey a message to a computer. Because high-level languages use everyday terms, they are easier for people to

learn than so-called low-level languages, which rely on words that only the computer can understand. But since a high-level language must be translated to a low-level language in order for the computer to understand it, higher languages such as SmartBASIC work relatively slowly.

HOME COMPUTER This is the term used to refer to the broad category of personal computers costing under a thousand dollars. Home computers tend to be used for game playing, word processing, money management topics and information retrieval, such as banking from home.

I

INPUT This is any kind of information that you type into ADAM. Input can be a word or line in a programming language such as SmartBASIC. Input can also be the response you give to ADAM after it encounters an input command.

INTEGRATED CIRCUIT A complete circuit that is typically formed by joining together minute electronic components on the surface of a single silicon chip. Known sometimes simply as ICs or chips, integrated circuits form the basis of ADAM's electronic brain.

INTERACTIVE COMPUTING The term given to the ability of computers to talk back and forth to users. Generally, with interactive computing, the user will type a command and the computer will respond. ADAM has the ability to perform interactive computing.

INTERFACE Any piece of machinery or computer program that connects the components of a computer system. In the case of ADAM, the cables between the screen and ADAM's memory console serve as an interface.

INTERPRETER A program inside ADAM that translates words from a computer language such as SmartBASIC into phrases the computer can more easily understand. Typically, the interpreter will translate each word in a SmartBASIC program, eventually converting the entire program into something ADAM can deal with.

J

JOYSTICK A computer instrument with a flat base and a stick or knob extending up from it, used to play computer video games. To control the position of a character on the screen, the joystick knob or stick is tilted in the desired direction. Also known as a "game paddle." ADAM comes equipped with two joysticks.

K

K Short for kilobyte, the scientific term for 1,024 (2^{10}) bytes. The term K is used to measure memory size.

KEYBOARD The most common form of input device for computers, a computer keyboard typically resembles the keyboard found on an ordinary typewriter. It has letters, numbers, punctuation marks and other symbols on it. In ADAM's case, the keyboard also contains keys dedicated to other purposes: the Smart Keys that change margins, delete, insert, alter screen color and perform other functions.

KEYPUNCH Used on older computers, keypunch is a machine that punches holes into rectangular cards. The pattern of the holes carries meaning and enables the user to input a program into a computer.

L

LIBRARY The term given to a group of computer programs, all of which are oriented to the same purpose. Educational programs, for example, in which children can learn reading, writing and other skills through the aid of the computer, would constitute a library of programs.

LIST The command in SmartBASIC that tells ADAM to list out all or part of a program. A listing, by comparison, is the copy of a program that ADAM provides after receiving a LIST command.

LOAD To transfer a program or other set of information into a computer's memory from another location, such as a disk drive or high-speed tape drive.

LOOP A series of programming instructions that make the computer repeat a process. One example of a loop would be a program designed to have the computer consider each number from 1 to 10, to raise each number to the power of 2, and to print the result on the screen.

M

MACHINE LANGUAGE The language that computers such as ADAM understand. Machine language uses a binary code, in which all letters, words, phrases, numbers and other characters are expressed using the base numbering system.

MAGNETIC DISK A soft, flexible plastic disk (typically encased in a hard sheath) on which information is stored electronically. Magnetic disks, known often simply as "floppy disks," have become the most popular storage medium for microcomputers.

MAGNETIC TAPE Tape, essentially identical to cassette tape, on which a magnetic substance has been applied so that information can be stored and retrieved. In the case of ADAM, the magnetic tape also has the ability to move swiftly through the tape drive, so it is called high-speed magnetic tape.

MAINFRAME A large computer, generally used by big corporations and governments, to carry out huge computing tasks. Mainframe is also the term used to describe the memory console of most computers, as separate from peripheral devices such as a printer, keyboard or screen.

MASS STORAGE SYSTEM A computer system for storing large quantities of programs, data or other information. With most small computers, mass storage takes the form of floppy disks used in conjunction with disk drives. ADAM, however, comes equipped with another method of mass storage: the high-speed magnetic tape and drive.

MEMORY The place in a computer system where information is stored, either temporarily or permanently. The information is translated into a binary code and is stored by the computer in the form of ones and zeros. Internal memory, as the phrase implies, is found inside the machine. Some of this internal memory has been permanently etched into the computer's hardware in the form of so-called read-only programs that can be read by the computer but not in any way changed. Other internal memory takes the form of a computer's working memory—the place where programs, data and other information are stored temporarily while being used. External memory, by comparison, usually consists of a floppy disk, magnetic tape or other memory device outside a computer, on which information is stored and called up as needed. Each piece of information is stored in a unique location in the computer's memory. Sometimes memory is simply called storage.

MENU A list of operations or procedures that a computer can carry out. A menu typically appears on a user's screen, and the user picks from the available choices to have the computer fulfill a specific task. Programs presenting the operator with a list of choices displayed on the screen are commonly referred to as menu-driven programs.

MICROCOMPUTER The smallest of computers, a microcomputer normally combines a microprocessor, a memory device such as a tape drive, an input device such as a keyboard, and an output vehicle such as a television screen to provide complete computing capabilities in a single package. Although smaller than minicomputers, microcomputers use the latest technology and cost hundreds of dollars less than minicomputers. The micro market is the fastest growing sector in the entire computer field.

MINICOMPUTER Midway between a microcomputer and a mainframe, a minicomputer is a medium-sized machine used typically by medium-sized firms for such diverse computing tasks as payroll and word processing. Since minicomputers are physically smaller than mainframes, they generally cost less than mainframes and offer less memory capability.

MODEM Short for MOdulator-DEModulator, a modem is a device that enables a home computer to transmit information electronically over telephone lines to communicate with other computers such as big, mainframe machines. The modem acts as the interface between the computer and the telephone, converting the binary-coded information inside the machine into electronic impulses that can travel over the phone lines.

MONITOR A screen, similar to that of a television set, designed especially to display the output of a computer.

N

NETWORK Any combination of computers, terminals, printers and monitors linked together to achieve a specific purpose such as inventory control in a department store. Networking often involves linking computers in separate, remote locations.

NIBBLE The pet name given by long-time computer enthusiasts to a half of a byte. Since a byte contains 8 bits, a nibble contains 4 bits. Generally, only half of a character, whether a number or letter, can be expressed by using a nibble.

NUMERIC VARIABLE A variable that has been defined beforehand to include only numbers. Numeric variables are in contrast to string variables, whose contents can be either numbers or letters.

O

ON LINE The term used to describe the situation when a terminal is hooked up to a "live" computer capable of doing all sorts of computing tasks. Since the user has established a direct link with the computer, we say that the user is working on line. This term may be used even if there is no single physical line linking the computer with its remote user.

OPERATING SYSTEM This is a permanent program built inside ADAM and other small home computers that acts as an internal housekeeper, supervising the running of applications programs and controlling the activity of various input and output devices (such as the keyboard and monitor). In ADAM's case, the operating system keeps track of what ADAM's central processing unit is doing, what the printer is up to, and

what is currently in memory, both working memory and mass storage outside the machine. The operating system is responsible for coordinating these diverse activities to make sure they run smoothly.

OUTPUT The broad name given to the display appearing on your television screen as a result of any work a computer such as ADAM may do. In addition, output refers to the printed results of a computing job, or to results that appear on a remote terminal.

OVERFLOW The term used to describe the situation in which the computer has ingested so much information that it stops dead in its tracks, unable to process any further information. In other words, if ADAM's memory is overflowing, it means it is filled up, sated with information, and cannot do any further work until some of its working memory is cleared.

P

PERIPHERAL EQUIPMENT Anything used in conjunction with a computer that is not technically the computer itself. In ADAM's case, the memory console would be considered the computer. The printer and monitor (as well as any other attachments you might buy, such as a modem or floppy disk drive) are considered peripheral equipment.

PORT Like a port on a ship, a computer port is nothing more than a hole in a computer. It is, more precisely, a point where you can make electrical attachments between the computer and other devices. More simply, a port is the place where you plug other devices into a computer.

PRINTER An output device used to type a copy of what appears on your screen onto a piece of paper. ADAM comes with a letter-quality printer, meaning that the printer is capable of typing out correspondence-quality prose virtually identical to that provided by a conventional typewriter.

PROCESSOR See *central processing unit.*

PROGRAM A series of instructions that the computer carries out in sequential order. Ordinarily, programs are written in programming languages such as SmartBASIC. A person who writes programs is called a programmer.

PRINTOUT A hard copy, on paper, of the output from the computer.

Q

QWERTY KEYBOARD The name given to a computer keyboard whose letters are arranged similarly to the way letters are arranged on most typewriters. If you think about it, the letters QWERTY shouldn't seem so foreign. They're just the first six letters on the left-hand side of the second row of the keyboard.

R

RANDOM ACCESS MEMORY Also known as RAM, random access memory is memory in which information can be both stored and retrieved. This is opposed to *read-only memory* (see also), from which information can be retrieved but which cannot be altered or used as a storage device. Random access memory is called random because information can be stored and retrieved extremely quickly; in a moment's notice, the computer can go to any random location in its memory. Random access memory is the type of memory used by ADAM as it processes programs and other information you may give it.

READ-ONLY MEMORY Known alternatively as ROM, read-only memory is information that has been permanently etched either into the circuits of a computer or onto a cartridge outside the machine. As such, read-only memory can be neither altered nor augmented. Information can be copied from read-only memory, but nothing new can be recorded on ROM.

RETURN KEY The key on the right side of ADAM's keyboard that serves the dual purpose of inserting carriage returns (if you're typing text), and of entering into the computer's memory individual lines of computer instruction (if you're writing a program in SmartBASIC).

RF MODULATOR An interface between a computer and a television that enables the computer output to be displayed on the television's screen. ADAM comes with an RF modulator. It's the little, rectangular metal box that you attached when you hooked up ADAM to the back of your television set.

S

SEMICONDUCTOR A material that is a better conduit of electricity than an insulator (such as wood), but not as good as a conductor (such as silver). Semiconductors such as silicon are vital to computers, for it is on tiny wafers of silicon that much internal electronic circuitry is etched.

SOFT COPY A copy of a text, data, program or other computer instruction that appears on a television screen or monitor while the computer is in use. As such, soft copies can't be picked up and toted away; they can only be viewed. This is opposed to a hard copy, as on paper, that can be picked up, edited with a pencil, and carried off.

SOFTWARE A fancy name for computer programs.

STATEMENT A line of a computer program, an individual command or instruction that you would give to ADAM.

STORAGE Another name given to ADAM's memory. The place where information—programs, data, and so on—can be stored and easily retrieved. Unlike physical storage (such as a warehouse where you might deposit your worldly possessions), ADAM's storage is all electronic. Everything that goes into ADAM's storage is recorded electronically inside tiny electrical circuits.

STRING A set of letters that are somehow connected in a logical sense. The word "ADAM," for example, would be considered a string. A sentence such as "ADAM is the best computer since sliced bread" would also be considered a string. Usually a distinction is drawn between string variables and numbers. ADAM can perform arithmetic only on numbers, not ordinary words as strings. It would make no sense to ADAM, for example, to multiply the following: cheeseburger × hotdog. The only thing ADAM could do with such edible strings would be to print them out on the screen, provided of course they appeared inside quotation marks in a PRINT statement.

SYNTAX The formal name given to the grammar of a programming language such as SmartBASIC. Just as in regular speech we say a person may have made an error in grammar if he speaks poorly, ADAM will tell you that you have made a syntax error if a command or other instruction you give it does not adhere to the grammar rules ADAM knows.

T

TAPE Short for magnetic tape. This is the storage medium that ADAM generally uses to record information. In ADAM's case, the magnetic tape looks and acts virtually identically to that used by a regular cassette recorder. Information is transmitted from ADAM's internal working memory to the tape, where it is recorded for retrieval at a later date.

TAPE DRIVE Similar to a conventional cassette recorder, a tape drive holds the magnetic tape that ADAM uses to store information. The drive is also responsible for spinning the tape to the appropriate point when a file is being retrieved. The drive also includes a so-called read/write head for reading from and/or writing information to a magnetic tape.

TERMINAL A machine that acts as a receiver and transmitter of information to a computer but that has no real computing power itself. Usually, the term is used in conjunction with telecommunications—the sending and receiving of information over phone lines. The terminal acts as a sort of ending point, where all information from a distant computer is received and viewed.

TEXT EDITOR Another term for a word processor, a text editor is a computer program designed to make writing and editing text extremely easy. With a text editor such as SmartWriter, it's possible to do all one's writing and corrections on a computer screen before printing out hard copy on paper. Only when the text is in finished form, after all insertions and deletions have been made, do you instruct the text editor to print your work.

V

VARIABLE An electronic pigeonhole that has a single identification but whose contents change. Specifically, a variable is a letter or word that is defined to represent a number, another letter or word, or to hold the results of an arithmetic computation. In the equation A = 7, A is a numeric variable. In the equation B$ = "Home Computer," B$ is a string variable. ADAM has been preprogrammed to understand both numeric and string variables.

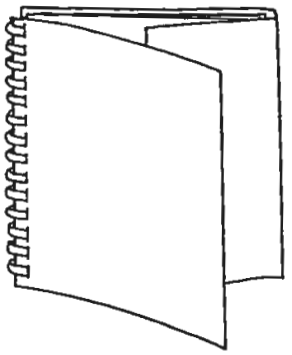
VERTICAL SCROLLING The process of moving lines of text up and down the screen so that additional text can be brought into view from below or above. Scrolling is necessary whenever a text is more than one screenful long. With ADAM it's possible to scroll up by striking the return key and to scroll down by using the cursor control keys.

VIDEO DISPLAY Another term for monitor, screen or CRT, a video display is a device designed to show output from a computer. Basically, you can think of a video display as the screen on your television set.

W

WAFER The term used to describe the tiny square sheets of silicon on which ADAM's electronic circuits are etched.

WORD PROCESSING A powerful computerized procedure for writing and editing text on a screen. With word processing programs such as ADAM's SmartWriter, it's possible to do all of your editing—deletions, insertions, moving of text—on the screen before printing out a final draft. It's also possible to store a body of text, which can then be retrieved at a later date for additional viewing or editing.



To save space while working at your computer, tuck the end flap inside the back cover. For easy reference on your bookshelf, fold over the end flap, tuck inside the front cover and position the book with the spine and title readily visible.