SAMS 22396

Commodore 64™ Trivia Data Base

Data Base and Trivia Game

James F. Hunter & Gregory L. Guntle

LIMITED SOFTWARE WARRANTY

This warranty applies only to the software portion of this product. If you purchased this book by itself, without the companion diskette or cassette tape, then this section does not apply to you.

For a period of ninety (90) days from the date of original purchase at retail, the warrantor, identified below, warrants this software to load and run as a basic program for the indicated microcomputer model, to be free from defects in material and workmanship and to be merchantable and suitable for its stated purpose for the period of this warranty. This warranty may not be enlarged except in writing, signed by warrantor. THE WARRANTOR EXPRESSLY DISCLAIMS ANY IMPLIED WARRANTY INCLUDING THE WARRANTY OF MERCHANTABILITY AND THE WARRANTY THAT THE SOFTWARE IS SUITABLE FOR ITS STATED PURPOSE AS OF THE DATE NINETY (90) DAYS FROM THE ORIGINAL PURCHASE OF THE SOFTWARE AT RETAIL.

In the event of defect, malfunction or failure of the software to conform with this warranty, the warrantor will repair or replace the software at no cost to you. For warranty service, you should return the software to the warrantor, **Howard W. Sams & Co., Inc., Attn: Sams Software, 4300 W. 62nd Street, Indianapolis, Indiana 46268.** Software received damaged as a result of shipping will require you to file a claim with the carrier. This warranty gives you specific legal rights and you may also have some other rights which vary from state to state.

THIS WARRANTY IS LIMITED SOLELY TO THE ABOVE AND THIS WARRANTY AND ANY WARRANTIES IMPLIED BY STATE LAW WILL APPLY ONLY FOR THE PERIOD SET FORTH. (SOME STATES DO NOT ALLOW LIMITATION ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.) THE WARRANTOR WILL NOT BE LIABLE FOR ANY LOSS, DAMAGE, INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY KIND, WHETHER BASED UPON WARRANTY CONTRACT OR NEGLIGENCE, AND ARISING IN CONNECTION WITH THE SALE, USE OR REPAIR OF THE SOFTWARE. (SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.) UNLESS OTHERWISE CONTRARY TO STATE LAW GOVERNING THE PURCHASE, THE WARRANTOR'S LIABILITY SHALL NOT IN ANY CASE EXCEED THE CONTRACT PRICE FOR THE SOFTWARE CLAIMED TO BE DEFECTIVE OR UNSUITABLE.

Commodore 64™ Trivia Data Base

First Edition
First Printing—1984

International Standard Book Number: 0-672-22396-1
Library of Congress Catalog Card Number: 84-51504

Edited by *Katherine Stuart Ewing*

Printed in the United States of America.

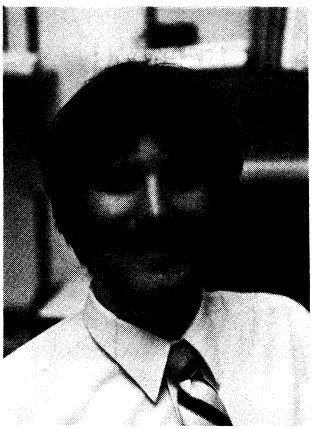
Commodore 64™ Trivia Data Base

by

James F. Hunter and Gregory L. Guntle

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

James F. Hunter is currently Director of Publishing for Howard W. Sams & Co., Inc. (ITT). A graduate of the University of California (Riverside), Jim is a veteran of seven years' experience in the personal computer field. In his spare time, he plays all board games with an enthusiasm and facility that sometimes astonish his opponents.



Gregory L. Guntle, a computer support specialist at Sams, is a 1983 graduate of Indiana University where he majored in computer science. He has worked actively with micros for the past six years. He also enjoys spending time in outdoors activities with his wife and family.

Preface

All computers, including micros, are designed to emulate processes of the human mind. It is, after all, we who have defined the tasks assigned to computers, with the goal of freeing ourselves from tedious and repetitious tasks which can be done more quickly and efficiently by an electronic device.

It is not unexpected that, after working with computers for a while, we begin to regard them as intelligent living entities. While not accurate technically, such an attitude can be of use in discussing what the programs in this book are designed to do. We need not be intimidated by the speed with which a computer can do repetitious tasks and calculations. Remember that, while doing those calculations, the computer need not be concerned with satisfying superiors on the job, raising children, paying bills, or trying to achieve goals set for itself, by itself. In short, a computer is not distracted by the state of being human.

For its part, the computer does not enjoy some of the very positive attributes of a human. It cannot think or feel or play. We can. Playing a data retrieval game against a computer would be no fun. The computer would always win. It is our own lack of perfection at manipulating and retrieving data that has accounted for the tremendous success of games like *Trivial Pursuit*. As we shall learn, the process of information storage and retrieval in a computer is exact and describable. Not so with us humans.

How many times have you heard someone say, "That reminds me of a story." Why? What is the mechanism which links one thought or event to another in the human mind? I don't have the answer, but I do believe that the lack of exact precision in describing those links can be a source of entertainment for people.

We are by nature curious. In the exercise of that curiosity, we amass tremendous quantities of information, some of which might not be of immediate or even long-term use. In an effort to justify

the acquisition and retention of such bits of knowledge, we at once name them trivia and proudly proclaim ourselves true fountains of useless information. In simpler terms, we try to express our belief that knowing things for their own sake is rewarding and fun.

The purposes of this book, and the included programs, are simple. The first purpose is to learn about the concept of a data base program on a computer, how it is developed, and how it works. The second and perhaps more important purpose is to take advantage of the given data base by using it as a pool from which to draw questions for the trivia random inquiry game program. Its third purpose is for you just to have fun. It is our hope that you will find both educational benefit and enjoyment in this book.

JAMES F. HUNTER

A Note To The Reader

The programs in this book were not written as applications software but as educational examples of what your personal computer can do. All of the programs have been tested and work on the machine configuration for which they were designed. The programs are unprotected. This means that you can modify them to better understand how they work or to fit a different machine configuration.

What Is a Combo Pack?

A Combo Pack, like this package, is a step beyond your average technical book. While most books give you programming examples through printed listings (which we do here), Combo Packs provide the book and the listings recorded on magnetic media, either disk, cassette tape, or both.

Every effort has been made to be clear, concise, and informative about how these programs and routines work. If you experience any difficulty with the software operations, the solution can be found in the book or in your computer manuals.

We are rather proud of the time and effort that went into preparing the Combo Pack. If you have purchased the Combo Pack and have enjoyed using it, let us know your thoughts. Your comments will be valuable in preparing future Combo Packs.

Loading Instructions

The cassette and disk accompanying this Combo Pack contain the program listings printed in the book.

To load a cassette file from the tape, perform the following steps:

1. Put cassette tape into the tape player.
2. To load the next file on the tape,
   Type: **LOAD**
   Press: <RETURN>
3. Follow the instructions presented on the screen, then the following will occur:
   The screen will blank out while the computer searches. When the file is found, the screen will print out the name of the file it has found and pause a few seconds. If the file is the one you want, do nothing. If the file is not the one you want, press <RUN/STOP> and <RESTORE> simultaneously. Then go back to step number 2.
4. Type **RUN** and press <RETURN> to run the program.

The following list shows the tape counter positions for the contents of the cassette tape. These numbers are approximate and may vary from recorder to recorder. They should, however, assist you in locating the programs you are searching for.

Tape Directory

Program Name	Counter Location
Data Base	0
Game	68
Sample File*	117

* Can only be loaded by the Data Base and Game programs.

Loading instructions — cont.

To load a disk file from the disk, perform the following steps:

1. Put the disk (label side up) into your disk drive and close the door.

2. Since the disk version is menu driven,
   Type: **LOAD "MENU", 8**
   Press: <RETURN>
   When the menu is loaded, type **RUN** and press <RETURN>

This program will automatically load and run all the files on the disk.

To load a specific file,
Type: **LOAD "FILENAME",8**
Press: <RETURN>
When the file is loaded, type **RUN** and press <RETURN>

Backing Up Your Master Disk Copy

Follow these steps in order to make one backup copy of the trivia master disk. The files listed below are those that you will need to transfer over to your backup disk:

MENU
TRIVIA DATA BASE
TRIVIA GAME

1. Format a blank disk. Take it out of your disk drive and label it "C64 - Trivia Back-up Disk".

2. Place the trivia master disk in your disk drive.

3. Type **NEW** <RETURN>
   LOAD "MENU",8 <RETURN>

4. After the program has been loaded, take out the trivia master disk and replace it with your backup disk.

5. Type **SAVE "MENU",8** <RETURN>

6. Follow steps 3 to 5 for transferring TRIVIA DATA BASE and for transferring TRIVIA GAME. Replace MENU with the name of the program that you wish to transfer.

Now you have a backup copy of the trivia master disk. Place the

master disk in a safe place. The sample file is still on your master disk.

Program Listing Conventions

The conventions used in listing programs in this book were selected for their clarity. The listings show the *keys to be pressed* rather than the character or result that is displayed on the screen. For example, a listing shows **{SHIFT/CLR-HOME}** instead of the reversed heart symbol. These listings are easier to type is you remember one simple rule:

Press The Keys Shown In Braces

In the listings, the following are keys to be pressed:

Notation:	**Press:**
{CLR-HOME}	CLR-HOME key
{SHIFT/CLR-HOME}	SHIFT and CLR-HOME keys simultaneously
{DA}	cursor down arrow key
{RA}	cursor right arrow key
{CTRL/BLK}	CTRL and BLK keys simultaneously
{CTRL/RVS ON}	CTRL and RVS ON keys simultaneously
{CTRL/RVS OFF}	CTRL and RVS OFF keys simultaneously
{COMM/WHT}	Commodore and WHT keys simultaneously
{COMM/BLU}	Commodore and BLU keys simultaneously
{*n* SP}	SPACE BAR n times ex. {5 SP} Press SPACE BAR 5 times

CONTENTS

Chapter 1
Introduction

In the 1940's, my father tuned pipe organs as a sideline, and he later worked on the first electronic organs. In the process of repairing and tuning those organs, he kept running into the fact that the twelfth root of two is a very significant number in understanding the tempered musical scale. And he needed to understand that scale thoroughly to do a good job of tuning. Because he worked principally as a motion picture projectionist at the time, he had ample opportunity to set about calculating by hand the twelfth root of two.

The process was simple, if time consuming. He would pick a number between one and two, multiply it by itself twelve times, and see how close the result was to 2.00000. If the result of the multiplications (all done by hand) was greater than 2.00000, he reduced the trial number. If it was less than 2.00000, then he increased the trial number.

Over a period of *years,* he was able to calculate the twelfth root of two to seven decimal places.

In 1975, I purchased a Hewlett Packard calculator. With a few keystrokes, I found the log of two, divided it by twelve, and took the antilog. In a matter of *seconds,* I'd obtained the same answer that it had taken my father years to arrive at. That speed in calculation is really what computers are all about. From the earliest modern computer, which was used by the U. S. Army to calculate mortar trajectories, to today's mainframe, mini, and microcomputers, the object has always been the same—to relegate repetitious and time consuming tasks to electromechanical (and now silicon technology) devices.

Repeated calculations to obtain a mathematical answer is popularly called *number crunching.* It was not long, however, after the advent of computers until other kinds of activities, which had previously been done by hand, were being done by computer. One

very obvious application is financial accounting, with its ledgers, balance sheets, T accounts, and profit and loss statements. One has but to remember the frustration of trying to balance a checkbook to understand the relief felt by accountants with the introduction of computerized bookkeeping.

As computers have become smaller, more affordable, and more powerful, other applications have been defined and implemented. Specifically, the applications which are of interest here are those currently being used on microcomputers, such as the Commodore 64. It is helpful to understand general groupings of those applications, so as to put the programs contained in this book and the accompanying cassette and disk in perspective. There are five main categories of microcomputer software, and each has several sub-categories:

Accounting
- General Ledger
- Accounts Payable
- Accounts Receivable
- Inventory
- Payroll

Productivity Tools
- Word Processor
- Spreadsheet
- Data Base
- Communications
- Graphics

Education
- Tutorial
- Skill Remediation
- Drill and Test
- Programmed Instruction
- Simulations

Entertainment
- Shoot-em-ups
- Strategy Games
- Fantasy Games
- Simulations

Utilities
  Programming Aids
  Communications
  Graphics

Looking quickly at the list above reveals that in this book we are dealing with an area of productivity tools called data bases. Before we begin construction of our data base, and subsequently use it for an amusing trivia game, we must describe and understand just what a data base is. And that's what is coming up in the next chapter.

Chapter 2

What Is a Data Base?

The term *data base* in itself is quite descriptive. A collection of information arranged in some non-random and accessible order is a data base. Your telephone white pages is a data base. The phone book gives multiple iterations of the same types of information for many listees—last name, first name, address, and telephone number. *The Joy of Cooking* cookbook is a collection of recipes, each of which has ingredients, and step-by-step directions for the preparation of food. It, too, is a data base. Given just these two examples, think about what other everyday sources of information could be regarded as data bases.

To better understand what electronic (computerized) data bases consist of and do, the following analogy to a commonly used manual system of maintaining a data base may be helpful. That system is the ever present 3 x 5 filing card system. Many such index card systems are the result of our desire to collect. Once our collection reaches a significant size, we need to have control of its contents. This control can be used to trade, to sell, to insure, to value, or for any number of other activities. For whatever reason, we definitely need to control the collection's contents.

Let us suppose that we collect cassette tapes of old time radio show broadcasts, and we want to be able to share them with friends. Our collection has grown to over 600 shows, and memory alone will not suffice to summon up the exact details of each show. Our friends ask if we have any Jack Benny shows. We know there are two, but where? Time for an index file! Time, indeed, for a data base!

Simply writing down the information about a show, in paragraph form for instance, quickly proves of limited use. For example:

> The Jack Benny show broadcast June 4, 1938. Guest stars include Edgar Bergen and Charlie McCarthy. The

show was sponsored by Lucky Strike, and is currently located in my upper-left hand desk drawer, on the Sony tape with the red and black label.

This card certainly has all the information we want, but after we have finished ten cards or so, we begin to file them. How? Alphabetically, of course. But alphabetically by what criteria? For starters, let's do it by the name of the show. What is the name of the show: "T" for the, "J" for Jack, or "B" for Benny? We obviously need some standard procedures. Also, when flipping through the cards, we need to be able to find the information on the show name quickly. Let's put the show name on a separate line at the top of the card.

The next thing we need to read on the card is the date of broadcast. Let's put it in the upper-right-hand corner. And the guests . . . second line, left-hand side. We are quickly designing a format for the information. Ultimately, it could end up looking like this:

Show title:	Bdcst Date:
Guests:	Sponsor:
Location:	Running time:
Additional Comments:	

At this point, let's digress for just a moment to point out another phenomenon resulting from the increasing use of microcomputers. It has been dubbed *computerphobia,* and it is often seen in the following form: the media has convinced us all that we must be "computer literate" if we are to survive economically and socially in the next decade. We also are "required," if we would be thought "good parents," to provide "computer literacy" for our children. Otherwise, perhaps we could be seen as impeding their growth and success potential as they enter a world controlled by computers. However, we sometimes feel inadequate (if not plain stupid) because we don't understand microcomputers. If we admit it by asking questions, people will then learn the worst—we really *are* stupid.

Fortunately, this is not at all the case. Most often, it is not the *concept* which we do not understand, it is the *jargon* used to describe computers and their uses. The following exercise may help to shed light on this point: set out below are two versions of a paragraph describing the use of our filing card system. The first

uses terms with which we are all familiar, and the description and concept are easily understood. The second, however, substitutes the terms which would be used to describe the same data base in a computer environment. By comparing the two, you see that (1) you can understand concepts of microcomputer usage, and (2) you are definitely *not* stupid.

Ordinary version

Now that we have the information layout, we can begin to fill out cards for each specific show. We can then file them alphabetically by show title. After all of the cards have been filled out, we can use the newly formed card file. If, as time goes by, we change our collection, we can remove cards, change cards, or insert cards to reflect those changes. We can also decide to file them alphabetically by another bit of information on each card, such as Guest Stars, and re-sort them for location using that new category.

Computerese version

Now that we have the format, we can begin to enter data for each specific show. We can then sort the data records by show title. After all of the records have been entered, we can access the newly formed data base. If, as time goes by, we change our collection, we can delete records, modify or edit records, or add records to reflect those changes. We can also decide to sort them alphabetically by another field, such as Guest Stars, and resave all records for access using the new key field.

In general terms, then, a computerized data base does the same things as a card system. So why bother with creating and maintaining such a data base? Because a computerized system can do many other things (far more quickly and easily) that the card system can't. For example, let us suppose we want to find a show that is exactly 28 minutes long. With the cards, we have to check each card by hand, or re-sort the cards by show length, from shortest to longest or vice versa. On the computer, we can search on the part of the card (field) which has that information until such a show is found, and then read (access) that whole record. Such sorts and searches using a variety of keystrokes are the backbone of an electronic data base.

Next, let us suppose we also establish a dollar value for each show, and enter that onto our format in its own location (field). A sophisticated data base allows us to add up all of the values, thus giving a total value for the collection at any point in time.

Finally, we shall assume that our insurance company wants limited information on each show in order to issue a policy on the collection. With the cards, we have to hand copy or electronically copy the cards. With an electronic data base, we could design a new format for printing the data, in columns for example, and summarize all of our shows on a few pages.

In summary, an electronic data base allows for the creation of data entry and output formats, and the actual entry, storing, retrieval, editing, deletion, searching, and sorting of records.

In addition to these features, our trivia data base will be complemented by a random inquiry program. Our data will consist of questions and one- and two-word answers, and the second program will randomly select questions from data entered into the data base program itself, compare our answers, and then score us on the speed and accuracy of our replies.

Chapter 3
Where Do We Begin?

Now that we understand something of what a generalized electronic data base is and what it does, we can begin to construct our own specialized random inquiry version of a data base. Beginning with Chapter 4, the method of this book is as follows: first, define the aspect of our data base program to be dealt with; and second, present and explain the BASIC language code that will achieve that result. As a tool for following the logical flow of both programs, we will use standard flowcharting techniques.

Such a process of program development is referred to as modular programming. Each task will have its own section of code, and when we put them all together at the end, we will have a program that meets our original design specifications.

Before we begin, however, let us review in more detail what it is that we want our program to do. First, we want to be able to enter trivia data questions and answers (in pairs). Next, we want to be able to edit (add, delete, alter) those entries until we are satisfied with the results. Finally, we want a second program to arrange for the computer to ask us those questions randomly, and give us an individual or comparative score for our efforts.

Perhaps a few comments on programming techniques and style would be helpful at this time. With regard to structure, the BASIC language can be something of a trap for the unwary. The necessity to access subroutines in other parts of our program could result in the creation of "spaghetti code" (a term meaning program code written in such a haphazard fashion that it "wanders" up and down and decreases program efficiency and legibility as it goes) if we are not careful.

Top down or structured programming is much to be preferred, and it means to start at the top of the program and work out the details in a logical, sequential manner. This requires that we have a very precise idea of how the logic of the program will work before we write a single line of program code.

The assignment of variable names within the program is also of great importance. Prepare a logical scheme for these variables by giving each variable a form which can be recalled easily. In other words, make the variable names mnemonic—a great help as we reuse the variables throughout the program.

REMark statements within the program help remind us of the function of modules, and aid other programmers (who might at some later date be working with the program) to understand the logic we are using.

Finally, we cannot always assume that a user will follow our directions, and we must therefore allow for circumstances in which input might not normally be expected, or it might take an invalid form. The process of accounting for such events is called "error trapping." This process is nothing more or less than anticipating inappropriate actions by the program's users, and preventing those incorrect actions from causing the program to fail, or "bomb" as we say in the trade.

We are now ready to build the skeleton of the program. Next stop, the flowchart (not Greenwich Village).

Chapter 4
Program Description

In the complete program listings, which appear in Appendices A, B, C, and D, you will note that this package actually consists of four different programs. In fact, there are two different versions of two programs, all delivered on the same cassette if you bought this package as a Combo Pack. On the tape (available separately as Howard W. Sams #26480), there is a tape version of the data base and the access routine. There is also a disk version (available separately as Howard W. Sams #26484) that will not execute on a tape based machine but is meant to be executed from a disk system. The disk version is discussed in greater detail in Chapters 11 and 12. (Disk version program listings and variable listings for main programs and subroutines are shown in Appendices C and D.)

One popular misconception (although *you* don't buy it for a minute!) about computers is that they can think. Computers cannot think. They can only be programmed to evaluate a certain situation according to certain guidelines, and then to perform certain calculations based on those guidelines. A specific example may be of some help.

Let us suppose that we want to create a program which receives as input from a user his sex, height, weight, and age. Then we want to program the computer to return a message as to whether the person who input the data is underweight, at a healthy weight (for him or her), or overweight. Before we can begin writing BASIC code, we must define the process by which the computer can output the appropriate response.

For a first pass, let us assume we have a chart of appropriate weights for adults. We can program the computer to retain that chart, and look up (based on the user input) an appropriate weight based on sex, height, and age. For now, let's just assume that a program exists that will accomplish that task. We now have,

in the computer's memory area, the target weight and the actual weight of our subject. Now we come to the kind of logical branching activities which computers can be programmed for and which lead people to believe that computers can indeed think. Actually, all that takes place are tests on the data with selection of the correct message based on the results of those tests.

For the sake of this example, let us further assume that anyone who is plus or minus 5% of his best weight is close enough, and will receive a positive message. If his weight is less than 95% of his target weight, he will get a "skinny" message, and if he is over 105% of his best weight, he will read a "fat" message.

The preceding description of program flow is wordy, awkward, and not easily understood at a glance. There must be a better way, and there is. What we have here is a sequence of tests, decisions, and actions, which can be represented nicely by a flowchart.

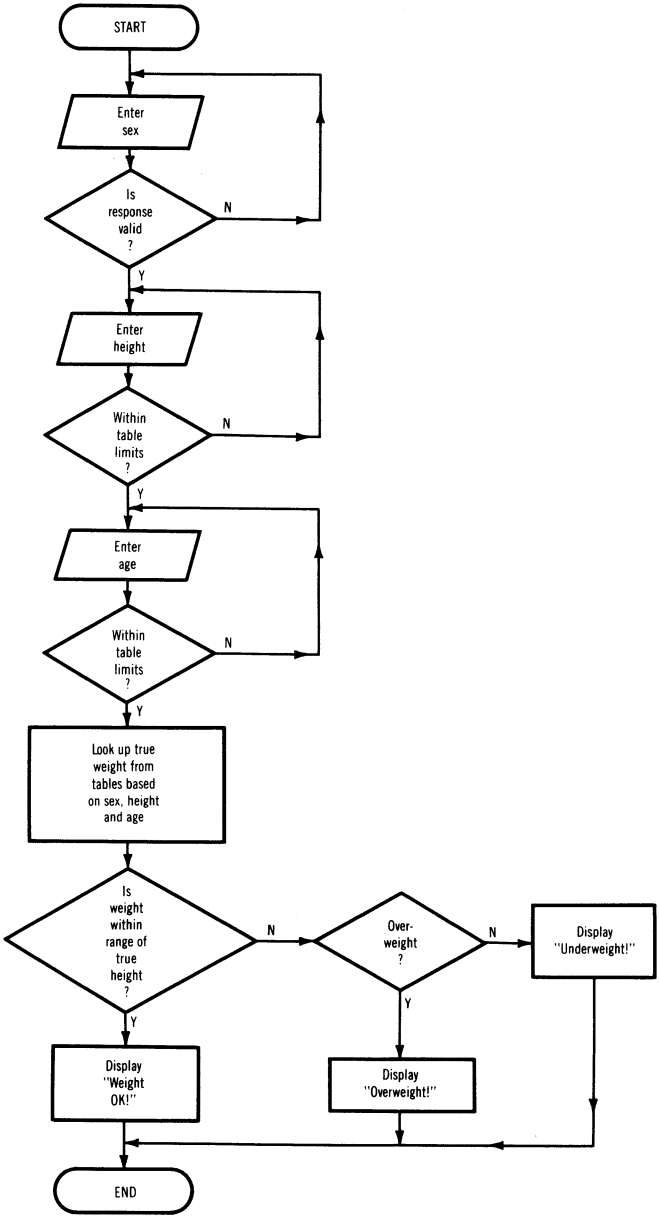


Fig. 4-1. Weight example program flowchart.

It is now clear at a glance that we receive input from a user, compare the actual and target weights, then choose one of three messages to print to the screen. Even in this simple example, the value of the flowcharting tool is evident. In more complex programs (such as we are describing here), flowcharting is an invaluable aid to understanding what is happening in the program. See Flowcharts 1, 2, 3, and 4 which fold out of the back of this book.

Using a flowchart, let us now look at the heart of our data base program (pull out Flowchart 1 for simultaneous viewing). From the standpoint of the user, his first decision will be to create a new data file, or load an existing one. If he chooses to load an existing file, he will be led through the process with screen messages; if he chooses to create a new one, he will be given four more choices. These lists of choices presented on the computer screen are called menus, and a program that has menu options available on the screen is said to be menu driven (and sometimes even "user friendly").

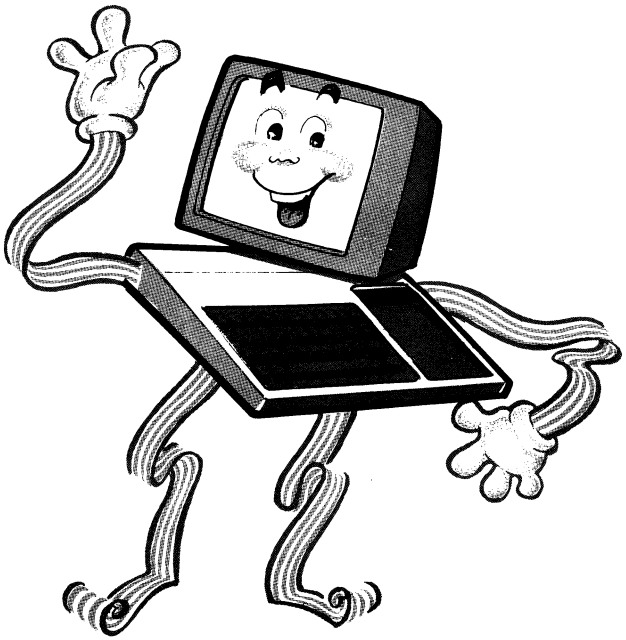


Fig. 4-2. Mr. User Friendly.

The BASIC code that will initialize our variables (they have to start somewhere) and present the first three choices on the initial menu looks like this:

```
10 DIM QA$(200):POKE 53280,13:POKE 53281,13
15 BLK$="{39 SP}":BLK$=BLK$+BLK$
20 FALSE=0:TRUE=NOT FALSE:GOTO 810



810 CNG=FALSE:NUMQ=0:PRINT"{SHIFT/CLR-HOME}{DA}{DA}
   {CTRL/BLK}{8 SP}**{2 SP}TRIVIA{1 SP}DATA{1 SP}BASE
   {2 SP}**"
820 PRINT"{DA}{DA}{DA}{2 SP}DO{1 SP}YOU{1 SP}WANT
   {1 SP}TO:{DA}{DA}":PRINT "{4 SP}1.{1 SP}CREATE
   {1 SP}A{1 SP}NEW{1 SP}TRIVIA{1 SP}FILE"
830 PRINT "{DA}{4 SP}2.{1 SP}LOAD{1 SP}AN{1 SP}EXISTIN
   G{1 SP}FILE":PRINT"{DA}{4 SP}3.{1 SP}EXIT{1 SP}DATA
   {1 SP}BASE{1 SP}PROGRAM"
840 PRINT"{DA}{DA}{4 SP}PLEASE{1 SP}SELECT{1 SP}(1-3)"
   :GOSUB 22
850 WAIT 197,64,64:GET SL$:IF SL$<"1" OR SL$>"3" THEN
   850
```

The four new choices are as follows:

```
        ** TRIVIA DB MAIN MENU **

        1. ADD QUESTIONS

        2. DISPLAY/EDIT QUESTIONS

        3. CHANGE FILES

        4. EXIT PROGRAM


        PLEASE SELECT (1-4)
```

Fig. 4-3. The Trivia DB main menu.

The code that produces this main menu follows:

```
1000 TN=NUMQ
1005 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{CTRL/BLK}{7 SP}**
   {2 SP}TRIVIA{1 SP}DB{1 SP}MAIN{1 SP}MENU{2 SP}**"
1010 PRINT"{DA}{DA}{DA}{7 SP}1.{1 SP}ADD{1 SP}QUESTION
   S":PRINT"{DA}{7 SP}2.{1 SP}DISPLAY/EDIT{1 SP}QUESTI
   ONS"
1020 PRINT"{DA}{7 SP}3.{1 SP}CHANGE{1 SP}FILES":PRINT"
   {DA}{7 SP}4.{1 SP}EXIT{1 SP}PROGRAM"
1030 PRINT "{DA}{DA}{7 SP}PLEASE{1 SP}SELECT{1 SP}(1-4
   )":GOSUB 22
1040 WAIT 197,64,64:GET SL$:IF SL$<"1" OR SL$>"4"
   THEN 1040
1045 POKE 55923,0:POKE 1651,ASC(SL$):SL=VAL(SL$):ON SL
    GOSUB 400,500,600,600
1050 GOTO 1005
```

Based upon which option is selected, there are three different subprograms that will be activated. On Flowchart 1, those choices will direct flow to B, C, or D. The following three chapters will deal with an evaluation and discussion of the flowchart for those options, and the resultant BASIC code that will achieve that flow.

Chapter 5
Adding Questions

Before reading further, pull out Flowchart 1 for simultaneous reference and see Appendix A.

Increment the counter . . . the phrase sounds like gobbledygook, but its meaning really is quite simple. The operation TN = TN + 1 indicates the next question will have a number that is one greater than the previous number. TN is a counter, and increment means add 1. Because we can only have up to 200 questions in one data base, we then test to see if we have exceeded that amount. If we have, then we let our user know. If not, we present a screen for input of the question and the correct answer.

But we are all human, and we might need to make some changes, because when we entered the question and answer, we made a mistake. We therefore offer the options of redoing the question and answer, saving it (and thus making it part of the current data file in memory only), or forgetting the whole thing and exiting the entry mode. If we choose to quit, we delete the current record but retain the rest of the saved records in memory. If we save what we have done, we set the CNG flag (simply a marker on the status of things) to TRUE, and save the work done to memory.

Subroutines are of great value to the programmer, because quite often a particular action is repeated throughout a program. Rather than copy the same code over and over, that action is identified as a "subroutine" and given a name. Then, when the action is required, that subroutine is invoked, or called, eliminating many lines of code.

In the code for this section, please note that there are a number of subroutine calls. The code is straightforward and simple.

The code for adding questions follows:

```
397 REM
398 REM   ADDING QUESTIONS
399 REM
```

```
400 PRINT"{SHIFT/CLR-HOME}":DE=FALSE
402 TT$="**{2 SP}ADDING{1 SP}QUESTIONS{2 SP}**":GOSUB
   200:GOSUB 60
405 TN=TN+1:IF TN <= 200 THEN 415
408 TN=200:PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}
   {CTRL/BLK}{4 SP}MEMORY{1 SP}IS{1 SP}FULL!!!":GOSUB
   22:GOSUB 50:RETURN
415 NM=TN:GOSUB 250:GOSUB 70:GOSUB 75
420 NR=2:FLG=0:SV=TRUE:EX=TRUE:SP=1305:GOSUB 100
422 IF FLG=1 THEN GOSUB 70:GOTO 420
424 IF FLG=3 THEN TN=TN-1:RETURN
426 IF ANS$=LEFT$(BLK$,LEN(ANS$)) THEN 420
428 Q$=ANS$
430 NR=1:FLG=0:SV=TRUE:EX=TRUE:SP=1665:GOSUB 100
432 IF FLG=1 THEN GOSUB 70:GOSUB 75:GOTO 420
434 IF FLG=3 THEN TN=TN-1:RETURN
435 IF ANS$=LEFT$(BLK$,LEN(ANS$)) THEN 430
436 ERR=FALSE:GOSUB 80:IF ERR THEN  GOSUB 284:GOSUB 75
   :GOTO 430
437 A$=ANS$
438 IF FLG=2 THEN GOSUB 194:GOSUB 22:GOSUB 22:CNG=TRUE
   :POKE 1981,32:GOTO 405
439 GOSUB 22:POKE 56253,0:POKE 1981,64
440 IF PEEK(198)=0 THEN 440
441 CH=PEEK(631):POKE 198,0
442 IF CH<133 OR CH>135 THEN 440
444 IF CH=133 THEN FLG=2:GOTO 438
446 IF CH=135 THEN TN=TN-1:RETURN
448 GOSUB 194:GOSUB 290:GOSUB 60:GOSUB 22:GOTO 440
```

There are two additional sections of code that are invoked within this portion of the program. It is desirable for the program to eliminate the articles "a", "an", and "the" from the answers we provide for the data base for two reasons. The first reason is that we are limited to one- and two-word answers. The second reason is that if we include articles in answers for the data base, and the program doesn't eliminate articles, the user would have to include those articles in his answers to get the answers correct. For example, if you enter "the Constitution" as the answer to a question in the data base, and the program does not eliminate "the", the trivia game player would have to answer the question "the Constitution", although "Constitution" is also correct. Lines 80-96 take care of eliminating the articles. (For complete lists of variable names for the programs in this book, see, as appropriate, Appendix A, B, C, or D.)

```
80 WD$="":WRDS=0:FOR I=1 TO LEN(ANS$)
82 IF ASC(MID$(ANS$,I,1))=32 AND WD$="" THEN 90
84 IF ASC(MID$(ANS$,I,1))=32 AND MID$(ANS$,I+1,1)="
   {1 SP}" THEN 90
86 IF ASC(MID$(ANS$,I,1))=32 OR I=LEN(ANS$) THEN WRDS
   =WRDS+1
```

```
87 IF WRDS>2 THEN ERR=TRUE:RETURN
88 WD$=WD$+MID$(ANS$,I,1)
90 NEXT I:FOR I=1 TO LEN(WD$):IF ASC(MID$(WD$,I,1))=32
    AND I<>1 THEN 93
92 NEXT I:RETURN
93 A$=MID$(WD$,1,I):T$=WD$
94 IF A$="THE{1 SP}" OR A$="AN{1 SP}" OR A$="A{1 SP}"
   THEN T$=MID$(WD$,I+1,LEN(T$)-I)
96 ANS$=T$:RETURN
```

The second section of code in this program portion handles the storing of questions and answers:

```
194 Q$=MID$(Q$+BLK$,1,74):A$=MID$(A$+BLK$,1,37):QA$(TN
   )=Q$+A$:RETURN
```

That, then, comprises the code for the ADD RECORDS section of the program. But, as noted earlier, nobody is perfect, and even after we have entered what we think is correct information, we sometimes will have to change it. Time to look to the editing process, and that is discussed in the next chapter.

Chapter 6
Editing Questions

This module is undoubtedly the most complex one in the data base program. Pull out Flowchart 1 for simultaneous reference, refer to Appendix A, and let's look at the logic.

First, we test to make sure our location in the list of questions is not before record number 1. If we are, we advise the user, and return to the main menu. If not, we assign our current position within the file to the variable ARP. We then display the current record question and answer, and offer it for modification. We have several options:

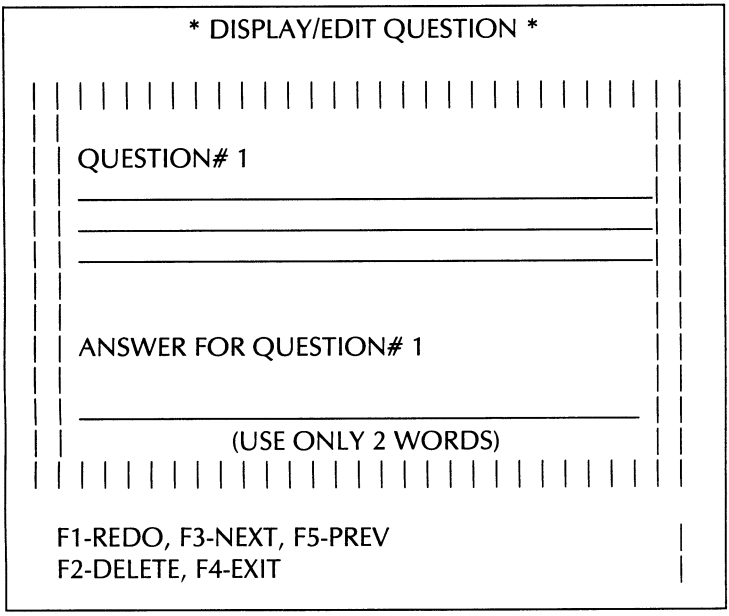


Fig 6-1. Display/edit screen.

The first option is to redo the question and/or the answer. Look

at the rest of the flowchart, and you can see that we have even more options. At this point, we can change the question, the answer, both, or neither. If we wish to change the question and/or the answer, we first erase the old entry, then replace it with the new information.

Notice that, as before, there is a flow option for "invalid responses." These are our error trapping routines. Now, let us return to option C in Flowchart 1.

```
287 REM
288 REM   REDO ROUTINE
289 REM
290 Q$=MID$(QA$(TN),1,74):A$=MID$(QA$(TN),75,37)
300 BOTH=FALSE:PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{2 SP}REDO:";LEFT$(BLK$,30)
310 PRINT"{7 SP}Q{1 SP}-{1 SP}QUESTION,{1 SP}A{1 SP}-
    {1 SP}ANSWER{6 SP}":PRINT"{7 SP}B{1 SP}-{1 SP}BOTH,
    {1 SP}";
312 PRINT"X{1 SP}-{1 SP}EXIT{10 SP}";
314 IF PEEK(198)=0 THEN 314
316 CH=PEEK(631):POKE 198,0
320 IF CH<>81 AND CH<>65 AND CH<>66 AND CH<>88 THEN 31
    4
322 IF CH=88 THEN RETURN
325 IF CH<>88 THEN GOSUB 190:CNG=TRUE
328 IF CH=81 THEN GOSUB 70:GOTO 334
330 IF CH=65 THEN GOSUB 75:GOTO 340
332 IF CH=66 THEN BOTH=TRUE:GOSUB 70:GOSUB 75
334 SV=FALSE:EX=FALSE:FLG=0:NR=2:SP=1305:GOSUB 100
336 IF FLG=1 THEN GOSUB 70:GOTO 334
337 IF ANS$=LEFT$(BLK$,LEN(ANS$)) THEN 334
338 Q$=ANS$:IF NOT BOTH THEN GOSUB 194:GOTO 290
340 SV=FALSE:EX=FALSE:NR=1:FLG=0:SP=1665:GOSUB 100
342 IF FLG=1 THEN GOSUB 75:GOTO 340
344 ERR=FALSE:GOSUB 80:IF ERR THEN GOSUB 284:GOSUB 75:
    GOTO 340
346 IF ANS$=LEFT$(BLK$,LEN(ANS$)) THEN 340
350 A$=ANS$:GOSUB 194:GOTO 290
```

The second option is to look at the next record. ARP is incremented by 1, then we test to make sure we have not passed all of the existing records in the file. If we pass both tests, it's back to display and editing. If not, we present an appropriate message, then return to the previous menu.

```
555 REM
556 REM  GET NEXT RECORD
557 REM
558 IF ARP<=TN THEN GOSUB 250:GOTO 530
```

```
560 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{3 SP}THERE{1 SP}ARE{1 SP}NO{1 SP}MORE
    {1 SP}QUESTIONS!{8 SP}"
562 PRINTLEFT$(BLK$,38);:GOSUB 22:D=600:GOSUB 280:ARP
    =ARP-1:GOTO 540
```

Likewise, we can also look at the previous record. We decrement the record position, check for the beginning of the file, then edit that record.

```
563 REM
564 REM  GET PREVIOUS RECORD
565 REM
566 IF ARP>=1 THEN GOSUB 250:GOTO 530
568 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{3 SP}CAN'T{1 SP}GO{1 SP}BELOW{1 SP}RECORD
    {1 SP}#1!{11 SP}"
570 PRINTLEFT$(BLK$,38);:GOSUB 22:D=600:GOSUB 280:ARP
    =ARP+1:GOTO 540
```

We may simply opt to delete a record altogether. We check to make sure that is the requested action (once it's gone, it's gone), then perform the action.

```
573 REM
574 REM  DELETE RECORD?
575 REM
576 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}";LEFT$(BLK$,39)
578 PRINT"{5 SP}DELETE{1 SP}THIS{1 SP}RECORD?{1 SP}(Y/
    N)":YF=FALSE:GOSUB 22
579 GOSUB 186:IF NOT YF THEN 540
580 IF TN-1=0 THEN QA$(ARP)="":TN=0:GOTO 500
582 IF ARP=TN THEN QA$(TN)="":ARP=ARP-1:GOTO 586
584 QA$(ARP)=QA$(TN)
586 TN=TN-1:CNG=TRUE:IF TN<1 THEN 500
588 NM=ARP:GOSUB 250:GOTO 530
```

Finally, we can choose to exit this module and return to the main menu.

```
552 IF CH=138 THEN RETURN
```

The code that produces the Display/edit screen is:

```
500 IF TN>0 THEN 510
502 PRINT"{SHIFT/CLR-HOME}{CTRL/BLK}{DA}{DA}{DA}
    {3 SP}THERE{1 SP}ARE{1 SP}NO{1 SP}QUESTIONS{1 SP}AN
    D{1 SP}ANSWERS":PRINT"{DA}{3 SP}IN{1 SP}MEMORY
    {1 SP}TO{1 SP}";
504 PRINT"DISPLAY/EDIT.":GOSUB 22:GOSUB 50:RETURN
510 ARP=1:DE=TRUE
```

```
520 TT$="**{2 SP}DISPLAY/EDIT{1 SP}QUESTIONS{2 SP}**":
    GOSUB 200
530 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";
    MID$(QA$(ARP),1,37):PRINT"{RA}";MID$(QA$(ARP),38,37
    )
535 PRINT:PRINT"{DA}{DA}{DA}{RA}";MID$(QA$(ARP),75,37)
540 NM=ARP:GOSUB 250:GOSUB 240:POKE 56241,0:POKE 1971,
    64:POKE 649,1
542 IF PEEK(198)=0 THEN 542
543 CH=PEEK(631):POKE 198,0
544 IF CH=133 THEN TT=TN:TN=ARP:GOSUB 290:GOSUB 194:TN
    =TT:GOTO 520
546 IF CH=134 THEN ARP=ARP+1:GOTO 558
548 IF CH=135 THEN ARP=ARP-1:GOTO 566
550 IF CH=137 THEN 576
552 IF CH=138 THEN RETURN
```

Chapter 7
Saving and Exiting

Now that we have finished with the data base, we need to leave the program cleanly, and allow for file updates as needed. Look at Flowchart 1 in the pullout section, as well as Appendix A, while we evaluate the final portions of the data base code.

First we check to make sure that some changes have been made to the active data base. If so, then you have a chance to save the file from memory to tape. Otherwise, we clear the screen to end the program.

If changes have been made, we ask if we want to save the changes. If so, we do; if not, we return to the main menu. All of these checks are designed to allow a user to "back out of" an erroneously chosen option without damaging the integrity of the data. The code for this section is:

```
600 IF NOT CNG AND SL=4 THEN POKE 53281,6:POKE 53280,1
    4:PRINT"{COMM/BLU}{SHIFT/CLR-HOME}":END
610 IF NOT CNG THEN 810
612 NUMQ=TN:YF=FALSE:PRINT"{SHIFT/CLR-HOME}{DA}{DA}
    {DA}{DA}{DA}{5 SP}SAVE{1 SP}CHANGES?{1 SP}(Y/N)":
    GOSUB 22
620 GOSUB 186:IF NOT YF AND SL=4 THEN POKE 53281,6:
    POKE 53280,14:PRINT"{COMM/BLU}{SHIFT/CLR-HOME}":
    END
630 IF NOT YF THEN 810
632 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{4 SP}PLEASE
    {1 SP}INSERT{1 SP}A{1 SP}CASSETTE{1 SP}WHERE"
633 PRINT"{4 SP}YOU{1 SP}WANT{1 SP}YOUR{1 SP}NEW
    {1 SP}QUESTIONS{1 SP}AND"
634 PRINT"{4 SP}ANSWERS{1 SP}TO{1 SP}BE{1 SP}SAVED.
    {DA}{DA}"
636 PRINT"{4 SP}THE{1 SP}SCREEN{1 SP}WILL{1 SP}GO
    {1 SP}BLANK.{2 SP}BUT":PRINT"{4 SP}DON'T{1 SP}PANIC
    !";
637 PRINT"{2 SP}THE{1 SP}SCREEN{1 SP}WILL":PRINT"
    {4 SP}RETURN{1 SP}AS{1 SP}SOON{1 SP}AS{1 SP}THE
    {1 SP}COMMODORE"
638 PRINT"{4 SP}HAS{1 SP}FINISHED{1 SP}SAVING{1 SP}THE
    {1 SP}FILE.":GOSUB 50:PRINT"{SHIFT/CLR-HOME}"
640 OPEN1,1,1,"TRIVIA{1 SP}FILE"
650 PRINT#1,NUMQ
655 FOR I=1 TO NUMQ
```

```
657 PRINT#1,MID$(QA$(I),1,74)
658 PRINT#1,MID$(QA$(I),75,37)
659 NEXT I:CLOSE 1
660 IF SL=4 THEN POKE 53280,14:POKE 53281,6:PRINT"
   {COMM/BLU}{SHIFT/CLR-HOME}":END
```

This completes the code explanations for the data base itself. Next, we shall look at the random inquiry and scorekeeping program (the game) using the same techniques employed in the preceding analysis.

Chapter 8
Beginning the Trivia Game

We can now begin an analysis of the second program in this package, the trivia game, which will utilize the data base created by the first program. Pull out Flowchart 2 for simultaneous viewing, see Appendix B, and we shall begin an evaluation of the flowchart and attendant code.

As was the case with the data base program, we must first initialize the variables. We then display the introduction screen, and ask if instructions are needed. This code appears as follows:

```
10 DIM QA$(200),QNU(30),CK(30),ND(8),NT(4):POKE 53280,
   13:POKE 53281,13
11 BLK$="{37 SP}":FALSE=0:TRUE=NOT FALSE
12 M=54272:FOR L=MTO M+24:POKE L,0:NEXT:GOTO 800

797 REM
798 REM   INTRODUCTION SCREEN
799 REM
800 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}
   {CTRL/BLK}{3 SP}WELCOME{1 SP}TO{1 SP}COMMODORE
   {1 SP}64{1 SP}TRIVIA{1 SP}GAME"
810 PRINT"{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{6 SP}NEED{1 SP}INSTRUCTION
   S?{1 SP}(Y/N)":YF=FALSE:GOSUB 22
820 GOSUB 22:GOSUB 186:IF NOT YF THEN 860
822 PRINT"{SHIFT/CLR-HOME}{CTRL/BLK}{9 SP}INSTRUCTIONS
   "
824 PRINT"{DA}{2 SP}THE{1 SP}OBJECT{1 SP}OF{1 SP}THE
   {1 SP}GAME{1 SP}IS{1 SP}TO{1 SP}GAIN{1 SP}AS":
   PRINT"MANY{1 SP}POINTS{1 SP}AS";
826 PRINT"{1 SP}POSSIBLE{1 SP}BY{1 SP}ANSWERING":
   PRINT"RANDOM{1 SP}TRIVIA{1 SP}QUESTIONS."
828 PRINT"{DA}{2 SP}EACH{1 SP}QUESTION{1 SP}IS{1 SP}WO
   RTH{1 SP}A{1 SP}STARTING":PRINT"VALUE{1 SP}OF
   {1 SP}25{1 SP}POINTS.";
830 PRINT"{2 SP}THE{1 SP}FASTER{1 SP}YOU":PRINT"ANSWER
   {1 SP}THE{1 SP}QUESTION{1 SP}THE{1 SP}MORE{1 SP}POI
   NTS";
832 PRINT"{1 SP}YOU{1 SP}CAN{1 SP}OBTAIN.{2 SP}AS
   {1 SP}TIME{1 SP}GOES{1 SP}BY,{1 SP}THE"
834 PRINT "POINT{1 SP}VALUE{1 SP}DECREASES.":PRINT"
   {DA}{2 SP}IF{1 SP}A{1 SP}PLAYER{1 SP}MISSES{1 SP}TH
   E{1 SP}ANSWER,"
```

```
836 PRINT"THE{1 SP}NEXT{1 SP}PLAYER{1 SP}IN{1 SP}LINE
   {1 SP}GETS{1 SP}A{1 SP}CHANCE":PRINT"TO{1 SP}ANSWER
   {1 SP}THE{1 SP}QUESTION.";
838 PRINT"{2 SP}ONCE{1 SP}ALL{1 SP}THE":PRINT"PLAYERS
   {1 SP}HAVE{1 SP}HAD{1 SP}A{1 SP}CHANCE{1 SP}TO
   {1 SP}ANSWER"
840 PRINT"THE{1 SP}QUESTION,{1 SP}THEN{1 SP}THE{1 SP}C
   ORRECT{1 SP}ANSWER":PRINT"IS{1 SP}DISPLAYED."
842 PRINT"{DA}{2 SP}THE{1 SP}GAME{1 SP}CAN{1 SP}BE
   {1 SP}PLAYED{1 SP}BY{1 SP}1-4":PRINT"PLAYERS.";
844 PRINT"{3 SP}GOOD{1 SP}LUCK!...HAVE{1 SP}FUN!"
846 PRINT"{DA}{CTRL/BLK}*{1 SP}ALL{1 SP}ANSWERS{1 SP}M
   UST{1 SP}MATCH{1 SP}EXACTLY!"
850 GOSUB 50
```

Next, we must load the data files from tape (or disk) prior to starting the game. First, all the questions are loaded into memory. Then, 30 questions are selected at random to be used for the game. The code for these activities follows (remember, subroutines are called by name, and can be found in Appendix B):

```
857 REM
858 REM   LOAD CASSETTE FILE
859 REM
860 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{2 SP}PL
   EASE{1 SP}INSERT{1 SP}THE{1 SP}CASSETTE{1 SP}THAT
   {1 SP}THE":PRINT"{2 SP}QUESTIONS{1 SP}AND";
862 PRINT"{1 SP}ANSWERS{1 SP}ARE{1 SP}STORED{1 SP}ON."
864 PRINT"{DA}{2 SP}IT{1 SP}IS{1 SP}GOING{1 SP}TO
   {1 SP}TAKE{1 SP}A{1 SP}WHILE{1 SP}TO{1 SP}LOAD":
   PRINT"{2 SP}THE{1 SP}FILE.{2 SP}PLEASE{1 SP}BE";
866 PRINT"{1 SP}PATIENT{1 SP}WHILE{1 SP}I":PRINT"
   {2 SP}LOAD{1 SP}THE{1 SP}FILE."
868 PRINT"{DA}{DA}{2 SP}DON'T{1 SP}PANIC!{2 SP}WHEN
   {1 SP}THE{1 SP}SCREEN{1 SP}GOES":PRINT"{2 SP}BLANK.
   {2 SP}IT{1 SP}WILL";
870 PRINT"{2 SP}RETURN{1 SP}AS{1 SP}SOON":PRINT"
   {2 SP}AS{1 SP}THE{1 SP}CASSETTE{1 SP}IS{1 SP}THROUG
   H{1 SP}BEING"
871 PRINT"{2 SP}LOADED."
872 GOSUB 22:GOSUB 50:PRINT"{SHIFT/CLR-HOME}"
874 OPEN1,1,0,"TRIVIA{1 SP}FILE"
876 INPUT# 1,NUMQ:FOR I=1 TO NUMQ
878 INPUT# 1,Q$
880 INPUT# 1,A$:QA$(I)=Q$+A$
881 NEXT I:CLOSE 1
```

Notice that, because this process is often time consuming, we display a "pacifier message" during the process to assure users that things are proceeding properly.

After we have selected our 30 questions (and their corresponding answers), we will pick double and triple point questions.

```
885 GOSUB 22:PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{3 SP}P
   LEASE{1 SP}WAIT...SELECTING{1 SP}QUESTIONS"
886 IF NUMQ<=30 THEN MN=NUMQ:GOTO 904
888 MN=30:FOR I=1 TO MN
890 WN=INT(RND(0)*NUMQ)+1:X=I-1:ERR=FALSE:GOSUB 500:
   IF ERR THEN 890
900 QNU(I)=WN:CK(I)=WN:NEXT I:GOTO 920
904 FOR I=1 TO MN
906 WN=INT(RND(0)*MN)+1:X=I-1:ERR=FALSE:GOSUB 500:IF E
   RR THEN 890
908 QNU(I)=WN:CK(I)=WN:NEXT I
920 DB=INT(MN*.25)+1:T3=INT(MN*.10)+1:FOR I=1 TO DB
924 WN=INT(RND(0)*MN)+1:X=I-1:ERR=FALSE:GOSUB 500:IF E
   RR THEN 924
926 ND(I)=QNU(WN):CK(I)=QNU(WN):NEXT I:FOR I=1 TO T3:C
   K(I)=0:NEXT:FOR I=1 TO T3
928 WN=INT(RND(0)*MN)+1:X=I-1:ERR=FALSE:GOSUB 500:IF E
   RR THEN 928
929 FOR J=1 TO T3:T(J)=CK(J):NEXT
930 FOR J=1 TO DB:CK(J)=ND(J):NEXT:ERR=FALSE:X=DB:
   GOSUB 500
932 IF ERR THEN 928
933 FOR J=1 TO T3:CK(J)=T(J):NEXT
934 NT(I)=QNU(WN):CK(I)=QNU(WN):NEXT I
```

Then we will find out who is playing by asking for the number of players and their names.

```
936 REM   NUMBER OF PLAYERS AND NAMES
937 REM
940 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{5 SP}NU
   MBER{1 SP}OF{1 SP}PLAYERS{1 SP}(1-4)":GOSUB 22
942 WAIT 197,64,64:GET NP$:IF NP$="" THEN 942
944 IF ASC(NP$)<49 OR ASC(NP$)>52 THEN 942
946 POKE55526,0:POKE 1254,ASC(NP$):NP=ASC(NP$)-48:FOR
   I=1 TO NP
947 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}"
948 PRINT"PLAYER{1 SP}#{1 SP}";I:PN$(I)="":INPUT "
   {DA}{DA}{2 SP}NAME";PN$(I)
950 IF PN$(I)="" THEN 947
952 PN$(I)=MID$(PN$(I)+BLK$,1,10):NEXT I
```

We are now ready to begin the game, so we display the game screen.

```
957 REM
958 REM   DISPLAY GAME SCREEN
959 REM
960 PRINT"{SHIFT/CLR-HOME}{DA}{CTRL/BLK}{10 SP}**
   {1 SP}TRIVIA{1 SP}GAME{1 SP}**"
961 PRINT"{DA}{CTRL/RVS ON}{40 SP}";
962 FOR I=1 TO 13:PRINT"{CTRL/RVS ON}{1 SP}";SPC(38);"
   {1 SP}";:NEXT
964 PRINT"{40 SP}";
966 FOR I=1 TO 5:PRINT"{1 SP}";SPC(38);"{1 SP}";:NEXT:
   PRINT"{20 SP}";
967 PRINT"{20 SP}{CTRL/RVS OFF}";
970 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{RA}QUESTION
   {1 SP}#":PRINT"{DA}{DA}{DA}{DA}{DA}{RA}ANSWER:"
972 FOR I=0 TO 36:POKE 55817+I,0:POKE 1545+I,120:NEXT
974 PRINT"{DA}{DA}{DA}{RA}{9 SP}(USE{1 SP}ONLY{1 SP}2
   {1 SP}WORDS){CTRL/BLK}"
976 PRINT"{DA}{DA}{DA}{RA}PLAYER{1 SP}#{10 SP}NAME:":
   PRINT"{DA}{RA}SCORE: {11 SP}NC/NQ:"
978 PRINT"{DA}{RA}{CTRL/BLK}{13 SP}F1{1 SP}-{1 SP}QUIT
   {1 SP}":AL=1:PL=1:FOR I=1 TO NP:SC(I)=0
980 NC(I)=0:NQ(I)=0:NEXT I:GOTO 200
```

In the next chapter, we look at the interrogation, evaluation, and scorekeeping functions.

Chapter 9
The Game Itself

For this section of the program, pull out Flowchart 2 and see Appendix B for reference. In Chapter 8, we displayed the game screen. Now, we must get one of the random questions previously loaded into memory. Then, we check to find out whether the question has been randomly designated as a double or triple value question.

Now, that we know what the question is worth in point value, we start the timer, and wait for the user to enter his answer. As time passes, the value of the question decreases. Once the <RETURN> key is pressed, we have to check for the correctness of the answer. If the answer is correct, and there is another player remaining, we go to the next player and display his current score. We then pick the next question, and display it.

Once the questions are used up, we must display the final score. This done, we ask for a replay and either start over or end the game with a pleasant message.

The following code checks to see whether the question is worth double or triple points:

```
200 GOSUB 64:NM=AL:GOSUB 450:TM=25:GOSUB 15:TP=25
202 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";
   LEFT$(QA$(QNU(AL)),37):PRINT"{RA}";MID$(QA$(QNU(AL)
   ),38,37)
203 PS=PL
204 PD=FALSE:FOR I=1 TO DB:IF ND(I)=QNU(AL) THEN PD=TR
   UE:GOTO 210
206 NEXT:PD=FALSE:FOR I=1 TO T3:IF NT(I)=QNU(AL) THEN
   PT=TRUE:GOTO 210
208 NEXT I:PT=FALSE
210 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{RA}";LEFT$(BLK$,37):GOSUB 75
212 IF PD THEN PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{RA}{10 SP}DOUBLE{1 SP}VALUE!":
   GOSUB 22:GOSUB 22
214 IF PT THEN PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{RA}{10 SP}TRIPLE{1 SP}VALUE!":
   GOSUB 22:GOSUB 22
```

The code that checks the answer for correctness, then goes to the next player and displays his score is:

```
220 TU=FALSE:EX=FALSE:SP=1505:GOSUB 100:PI=TM
222 IF EX THEN 600
224 IF TU THEN 240
226 IF PD THEN PI=PI*2
228 IF PT THEN PI=PI*3
230 GOSUB 80:ANS$=MID$(ANS$+BLK$,1,37):CA$=MID$(QA$(QN
    U(AL)),75,37)
232 IF CA$=ANS$ THEN 250
240 GOSUB 30:GOSUB 30:D=300:GOSUB 60:NQ(PL)=NQ(PL)+1:P
    L=PL+1:IF PL>NP THEN PL=1
242 IF PL<>PS THEN TP=12:GOSUB 15:GOSUB 64:GOSUB 75:
    GOTO 220
244 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{RA}THE{1 SP}CORRECT{1 SP}ANSWER{1 SP}IS:":
    GOSUB 30
245 PRINT"{DA}{RA}";MID$(QA$(QNU(AL)),75,37):D=1000:
    GOSUB 60
246 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{RA}ANSWER{19 SP}"
248 GOSUB 75:GOTO 260
250 GOSUB 22:GOSUB 22:SC(PL)=SC(PL)+PI:NC(PL)=NC(PL)+1
    :NQ(PL)=NQ(PL)+1
252 GOSUB 64:D=500:GOSUB 75:GOSUB 60
260 AL=AL+1:IF AL>MN THEN 280
262 PL=PS+1:IF PL>NP THEN PL=1
264 PD=FALSE:PT=FALSE:GOTO 200
280 D=400:GOSUB 64:GOSUB 60:GOTO 750
```

Once the game begins, we may wish to quit (when the score is terribly one-sided, for example). We enter **F1** to exit the game early:

```
597 REM
598 REM   QUIT GAME EARLY?
599 REM
600 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{RA}{9 SP}ARE{1 SP}YOU{1 SP}SURE?{1 SP}(Y/N
    )":GOSUB 22
610 YF=FALSE:GOSUB 186:IF NOT YF THEN PRINT"
    {CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {RA}";
620 IF NOT YF THEN PRINT"{13 SP}F1{1 SP}-{1 SP}QUIT
    {7 SP}":TP=PI:GOSUB 75:GOTO 220
```

Notice that, as always with an ending decision, we check for a confirmation.

This completes the code for the actual game. When we put it all together, it should (and does) work. In the next three chapters, we shall document the actual use of these two programs.

Chapter 10

Using the Cassette Data Base

As noted earlier, the trivia data base is the program used to store your questions and answers. This chapter demonstrates how easily the data base program can be to use. (Just a note before we begin: you will need to have blank cassettes handy at all times, and they should be labelled in some organized way. For example, Trivia Files #1, Trivia Files #2, etc., or even Cassette #1, Cassette #2, etc., will work. The reason for the blank cassettes is that the program (for memory convenience) stores 200 questions and answers on each cassette and if you have more than 200 questions you'll need to store the rest on a different cassette.)

The data base is totally menu driven. This means that you are able to select from several choices the process that you want to perform on the data base. Once the program is run, you are presented with an initial menu as shown in Fig. 10-1.

From this menu you select whether you want to create a new trivia file, load and work on a trivia file that has already been created with this program, or exit the data base program. Because this is your first time using the data base, you'll want to select option #1, CREATE A NEW TRIVIA FILE, but before doing that, let me explain the options. (Note that option 3, EXIT DATA BASE, is described in Chapter 7.)

CREATE A NEW TRIVIA FILE

This option allows you to create a new file. Any file can contain, at most, 200 questions and answers. It is because of memory and speed considerations that we arrived at the limit of 200. Go ahead and select this option now, then you are presented with the Trivia DB main menu.

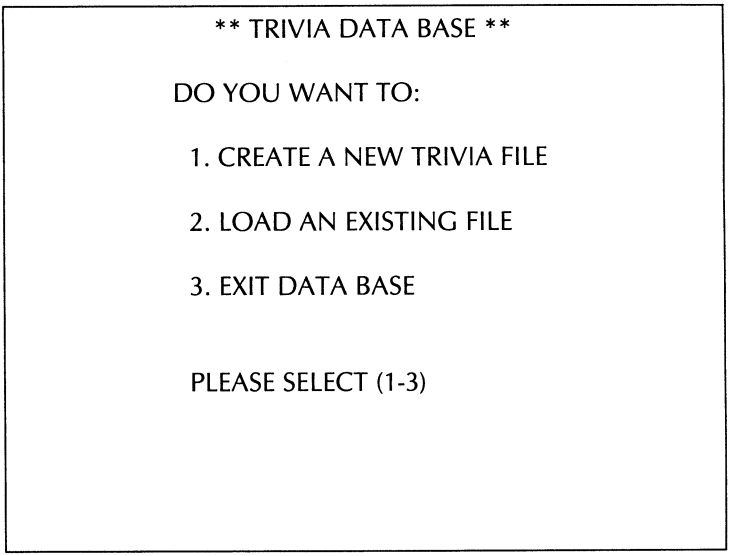


Fig. 10-1. The Trivia Data Base initial menu.

LOAD AN EXISTING FILE

This option allows you to load a trivia file into memory, so you are able to add new questions to the file or change questions and answers. Once this option is chosen, a message is displayed instructing you to insert your cassette file into your tape player or a disk in the drive depending upon which version of the program you are using. After you choose whether to create a new file or load an existing file, the screen will show the menu in Fig. 10-2.

Let's look at each of the four selections given in the main menu.

1. Add Questions

This is the first thing you need to do when creating a new file. This selection allows the user to enter new questions and answers to the trivia file. To let you know how many questions are in the file, the current question number is always displayed while you are adding new questions. The screen for adding the questions and answers is shown in Fig. 10-3.

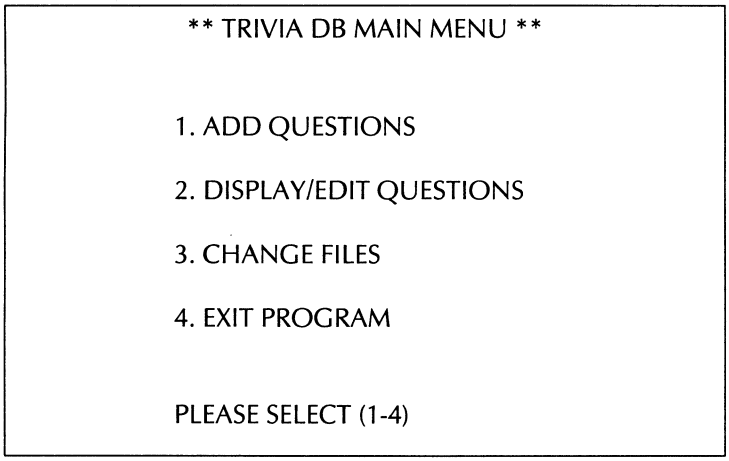


Fig. 10-2. The Trivia DB main menu.

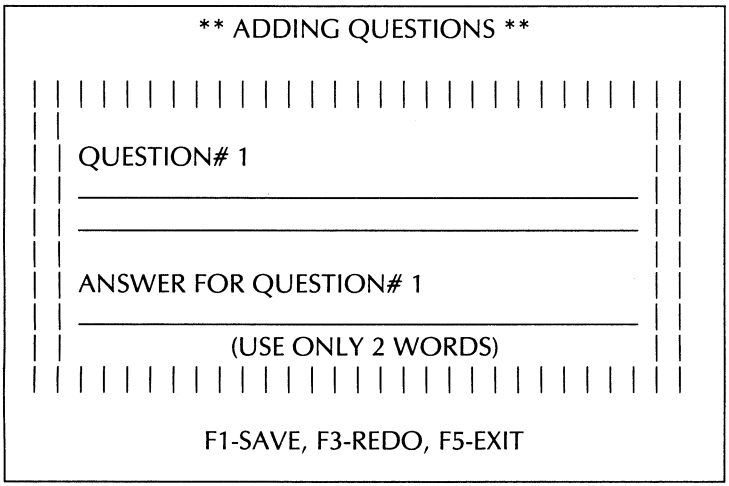


Fig. 10-3. Adding questions screen.

First take a look at the bottom of the screen in Fig. 10-3. The bottom of every screen contains valuable information. This screen contains only three major keys. They are **F1**, **F3**, and **F5**. These are the function keys that are located along the right side of the keyboard. So by pressing the **F1** key, the current question and answer

is saved into memory. **F3** allows you to redo or change the questions and answers. If the cursor (the little black rectangle that appears on the question field) is on the last line of the screen, by the three major keys, then **F3** generates another display asking what you want to change. You can then choose to edit the question, answer, or both. If you press **F3** while the cursor is on either the question field or the answer field, you must reenter both the question and the answer. The data base doesn't allow you to enter a blank question or a blank answer. Also, you may press any one of the keys displayed at the bottom of the screen at anytime. By pressing the **F5** key, you exit the ADD QUESTION routine and also erase anything that is on the screen at the time of exiting. You are then returned to the Trivia DB main menu (Fig. 10-2).

Now you can enter your first question. When you are finished with your question, press the <RETURN> key to move the cursor down to the answer field. The answer cannot be more than two words. If it is, the computer beeps and tells you to reenter the answer. Once the user has typed in an answer, you can use **F1** to save it or press <RETURN> to move the cursor down to the command line at the bottom of the screen. Again you may press any one of the three command keys at any time. Once the question and answer have been saved, you are given another blank screen to add more questions and answers.

When the cursor is on the same line as the command keys, you have to press one of the three keys displayed. If the user presses **F1**, the record is saved to memory, the array counter is incremented, and another blank record is displayed. If the user presses **F5**, the current question and answer on the screen are erased, and the user is returned to the Trivia DB main menu (Fig. 10-2). If you press **F3**, you are given the REDO command line presented at the bottom of the screen (Fig. 10-4).

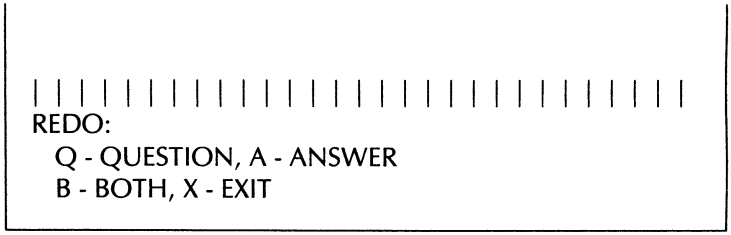


Fig. 10-4. REDO command line.

Once at this point, you can select which part you want to redo. Either the question, the answer, or both at once. When you are finished editing the record, press **X** to exit and return to the **F1**, **F3**, etc., commands. You will still need to save the record when you are through editing it.

2. Display/edit Questions

This routine allows you to display as well as edit (make changes) to any question or answer in the file. When you press 2 from the main menu, you are presented with the Display/edit screen (Fig 10-5).

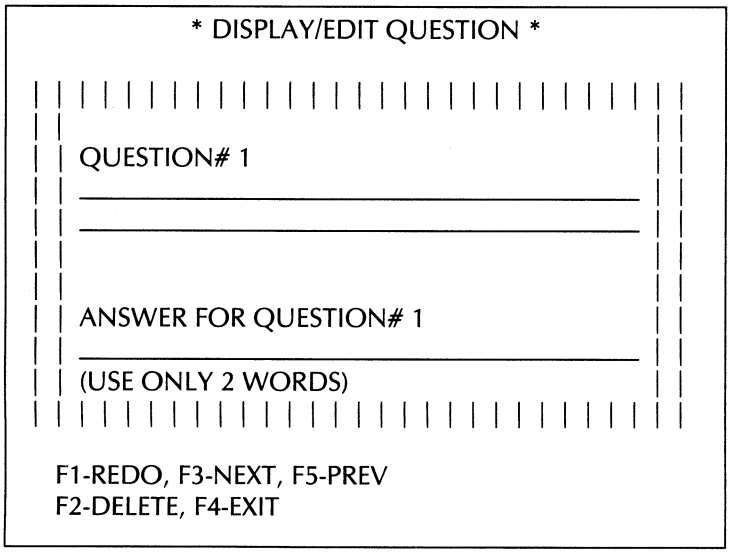


Fig 10-5. Display/edit screen.

Again, as shown in Fig 10-5, at the bottom of the screen there is another command line. Here, there are five command keys to use. **F1** displays two rows of commands that allow the user to change the question, the answer, or both, as in Fig. 10-4. You can then choose what you want to do. **F3** displays the next question and answer. **F5** displays the previous question and answer. **F2** (entered by pressing the **SHIFT** and **F1** keys) asks you if you are sure that this is the question and answer you want to delete. If you

press **Y** to the prompt, the question and answer is deleted from your file. **F4** (entered by pressing the **SHIFT** and **F3** keys) exits this routine and takes you to the Trivia DB main menu.

3. Change Files

This selection allows you to quit working on the file currently in memory and load in a different file. If there were any changes made to the file in memory, then you are asked if you wish to save the changes made to the file. You are prompted with this question: SAVE CHANGES? (Y/N). If you press **N**, everything (including the changes) will be thrown away. If you press **Y**, you are prompted to insert a cassette in the tape recorder or a disk in the drive depending upon which version of the program you are using. Once the file in memory has been saved or erased, you are returned to the very first screen (Fig 10-1), and you choose from that screen what you want to do.

4. Exit Program

This selection allows you to exit the data base program. Again, if there is any file in memory and changes (additions, deletions, or editing) were made to the file, you are asked if you wish to save the file that is currently in memory. If you elect not to save the file in memory, everything in memory is thrown away. *Always exit the program through this selection. If you don't, none of the changes or new questions can be saved.*

Now that you have an understanding of how to enter and edit questions, try putting some questions and answers in a file so you can have some questions and answers with which to play the trivia game. If you really can't wait to play the game, go ahead and load the sample questions contained on side 2 of the cassette.

NOTE ON USING THE SAMPLE FILE

There are a few mistakes in the sample file. They have been made on purpose to provide you with a file on which to practice using the data base. Remember to save the new file when you are finished.

Chapter 11

Playing the Cassette Trivia Game

In the last chapter, you learned how to enter questions and answers into your trivia data base file. This chapter uses those questions you entered into the file and allows you to play a trivia game. For those of you who did not store any questions and answers using the data base, you may use the sample file located on side 2 of the cassette.

INSTRUCTIONS

The game itself is much like any other trivia game. You are asked random questions and you have to answer them. The object of the game is to gain as many points as possible by answering random questions. The faster you are able to answer a question, the more points you will receive. The point value for each question starts at 25 points, but as time goes by the point value decreases. So, you see, it is to your benefit to answer the questions as fast as possible. There will be times when a question will be worth double and even triple points. One note: *the answer you type must match the correct answer exactly (except that you may add "a", "an", or "the" to one-word answers).* If the answers do not match, your answer will not be counted as correct. The game can be played by one to four players. If a player misses a question, then the next player in line gets a chance to answer that question, although the point value is reset to 12 points (24 if double value question, 36 if triple value) instead of the original value of 25 (50 if double value question, 75 if triple value). If everyone has a chance to answer the question and no one answers it correctly, the correct answer is displayed. Every time someone gets the correct answer, a high beeping sound is made. When some-

one misses, they hear a lower beeping sound. The game keeps track of each player's number, his/her name, total points scored, the number of questions each player has answered correctly, and the number of questions each player has had a chance to answer. At the end of the game, the players' scores are shown in descending (from highest to lowest) order.

RUNNING THE GAME

The first screen you see is the introduction screen (Fig. 11-1). It welcomes you to the trivia game and asks if you need instructions. The trivia file is then loaded into memory. Once all the questions are in memory, the program randomly selects 30 questions to be used for the game. A percentage of the questions is selected to be worth double the point value, and a few of the questions are selected for triple value. Once all the questions have been scrambled, you are asked how many people plan to play and to enter their names into the computer. Now, the game is ready to begin.

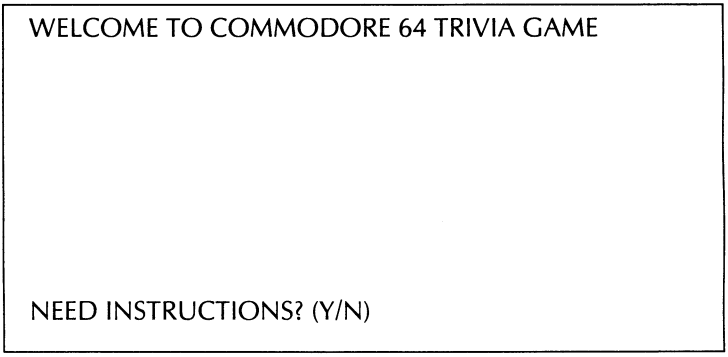


Fig. 11-1. The introduction screen.

THE START

After everyone has entered his/her name, the game screen is displayed (Fig. 11-2). Notice in the upper-right hand corner of the boxed-in screen, there is a timer. This timer also shows how much the question is worth. As time goes by, this timer decreases.

When you press <RETURN> after entering your answer, the timer freezes at the point value shown. This is the number of points you will receive if your answer is correct. At the bottom of the screen is the status of the current player. It shows the player number, name, score, the number of correct answers (NC) and the number of questions (NQ) that player had a chance to answer. Once you press <RETURN>, the program checks your answer against the correct answer. If you are right, you receive the number of points shown in the upper-right hand corner. Otherwise, the next player gets a chance to answer the same question. The correct answer is displayed if everyone has a chance to answer the question, and they all miss.

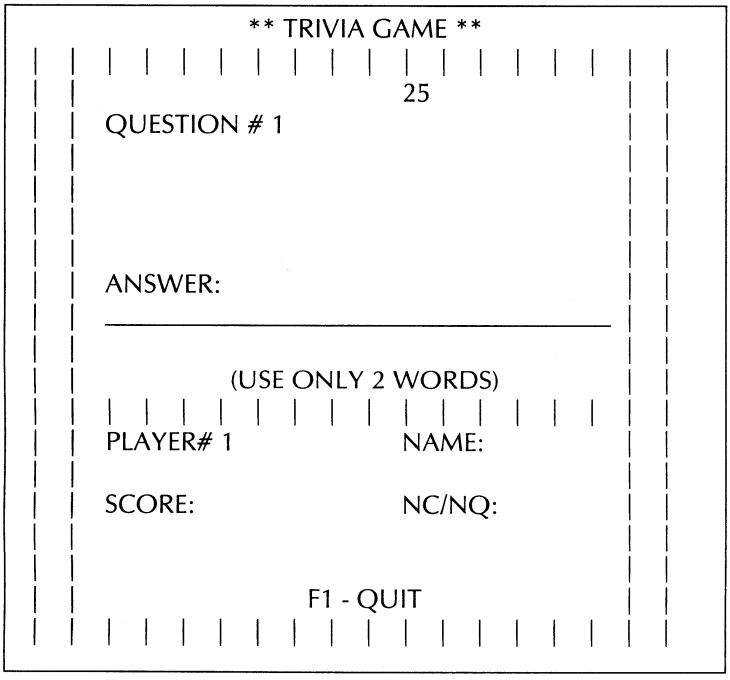


Fig. 11-2. The game screen.

EXITING THE GAME

The game normally ends when all the questions have been asked. Then there is a status screen displayed that shows all the players' scores in descending (highest to lowest) order. You are then asked if you wish to play another game. If you don't, the program exits. Otherwise, you start all over again by loading in another file, etc.

If you want to end the game before all the questions have been asked, press **F1**, and the program will ask if you are sure you want to exit. If so, all the players' scores are shown with the highest score shown first. If you do not wish to exit, the game continues with the same question and the same point value that the question was at before you pressed **F1**.

The main point behind the trivia game is for you to enjoy the game, while at the same time learning new and different trivia questions. Have fun!

Chapter 12

Using and Playing the Disk Versions

This chapter explains the use of the disk versions of the data base and trivia game. To load the trivia menu, follow the loading instructions in the front of the book. Once you load and run the Menu program, the menu selection screen (Fig. 12-1) is displayed.

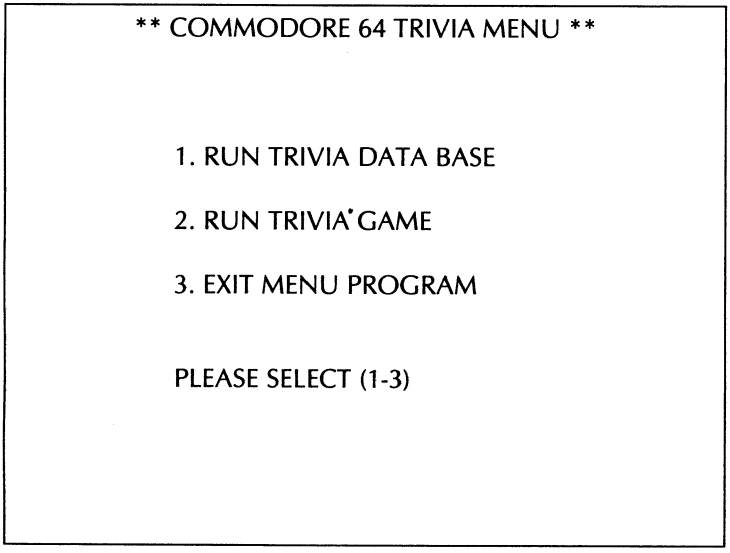


Fig. 12-1. The Disk Trivia Data Base.

To use the trivia data base, press **1**, to play the trivia game, press **2**. To exit the menu program, press **3**. The code that produces this menu follows.

```
7 REM
8 REM   MENU PROGRAM
9 REM
10 POKE 53280,13:POKE 53281,13
15 READ MN$(1),MN$(2),MN$(3)
20 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{CTRL/BLK}{4 SP}**
   {2 SP}COMMODORE{1 SP}64{1 SP}TRIVIA{1 SP}MENU
   {2 SP}**"
30 PRINT"{DA}{DA}":FOR I=1 TO 3:PRINT"{DA}{DA}{6 SP}";
   I;"{1 SP}";:IF I<>3 THEN PRINT"RUN{1 SP}"+MN$(I)
40 IF I=3 THEN PRINT MN$(I)
50 NEXT
60 PRINT"{DA}{DA}{7 SP}PLEASE{1 SP}SELECT{1 SP}(1-3)"
70 WAIT 197,64,64:GET SL$:IF SL$="" THEN 70
80 IF VAL(SL$)<1 OR VAL(SL$)>3 THEN 70
90 POKE 1733,ASC(SL$):IF VAL(SL$)<>3 THEN 95
91 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{3 SP}EXI
   T{1 SP}PROGRAM?{1 SP}(Y/N)"
92 WAIT 197,64,64:GET CH$:IF CH$<>"Y" AND CH$<>"N"
   THEN 92
93 IF CH$="Y" THEN POKE 53280,14:POKE 53281,6:PRINT"
   {SHIFT/CLR-HOME}{COMM/BLU}":END
94 GOTO 20
95 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{1 SP}LOAD
   {1 SP}";MN$(VAL(SL$));"?{1 SP}(Y/N)"
100 WAIT 197,64,64:GET YN$:IF YN$<>"Y" AND YN$<>"N"
   THEN 100
110 IF YN$="N" THEN 20
120 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}LOAD"CHR$(34)MN$
   (VAL(SL$))CHR$(34)",8"
130 PRINT"{CLR-HOME}":POKE 198,4:POKE 631,13:POKE 632,
   82:POKE 633,213:POKE 634,13
200 DATA TRIVIA DATA BASE,TRIVIA GAME,EXIT MENU PROGRA
   M
```

USING THE DISK DATA BASE

The disk data base is very similar to the cassette version. In fact, the menus and all the other screen options displayed in the cassette version are used in the disk version. There are, however, some significant differences between the two versions. The major difference between the cassette and this version is the I/O (input and output) routines. These are the routines that allow the data to be saved to and read from the disk. In the cassette version, everything (all the questions and answers) are kept in memory until you change files or exit the program. Then the questions and answers are saved to your cassette in sequential order. In this version, at most only one question and answer is in memory at a time and the disk uses RELATIVE files for faster loading and saving of the data.

Also, keep the disk in your disk drive at all times while running the data base program. If you don't, some unexpected results can occur, like losing some data or even losing the whole file.

The red disk light on your disk drive will remain on at all times once the file has been opened. *Don't open the drive door while the light is on.*

Every disk you use must be formatted. For information on formatting a new disk, refer to the Commodore *User Manual* for your disk drive.

Use only one disk per trivia file. The program is set up to fill a whole disk with one file. If you put more than one trivia file on a disk, the computer will eventually indicate a DISK FULL error, and you will lose some data. Each disk can contain 720 questions and answers, if there are no files other than trivia files on the disk.

PLAYING THE DISK TRIVIA GAME

The disk trivia game is very much like the cassette game. The only difference is the way the questions and answers are loaded.

The trivia disk must remain in your disk drive until the questions are loaded. Then you can remove the disk. Only 30 random questions are loaded.

Have fun!

Chapter 13
In Conclusion

Well, that's it. We have provided sample questions of the trivia game on side 2 of the tape and on the disk. We really strained our brains for those questions, but now it is up to you. If you have a copy of *Trivial Pursuit* (or some such game), you could enter some selected questions and play, with the computer keeping score. If not, there are some other uses for the data base entry program and game program.

If you have students who need to drill on facts (answers no longer than two words), they could have the questions entered (by you?), then practice with the computer's assistance. If you are studying for a professional exam, the same sort of assistance is available for you.

Whether you use these two programs for entertainment or education, it is our hope that you will gain from their purchase. Also, a thorough examination of the code structure and subroutines will be of great assistance in any other data base type programs which you might wish to write for your Commodore 64.

Watch for other entertaining and instructional programs from us for your Commodore 64.

Appendix A

The Cassette Data Base

PROGRAM LISTING

```
5 REM
6 REM  CASSETTE VERSION - DATA BASE
7 REM
10 DIM QA$(200):POKE 53280,13:POKE 53281,13
15 BLK$="{39 SP}":BLK$=BLK$+BLK$
20 FALSE=0:TRUE=NOT FALSE:GOTO 810
22 M=54272:FOR L=MTO M+24:POKE L,0:NEXT:POKEM+5,9:
   POKEM+6,0:POKEM+24,15
24 POKEM+1,25:POKEM,30:POKEM+4,33:FORD=1 TO 150:NEXT:
   POKEM+4,32
26 POKE M+24,0:RETURN
50 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{5 SP}PRESS{2 SP}<RETURN>{2 SP}TO
   {1 SP}CONTINUE"
55 WAIT 197,64,64:GET RT$:IF RT$<>CHR$(13) THEN 55
58 RETURN
60 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}";LEFT$(BLK$,39)
61 PRINT"{4 SP}F1{1 SP}-{1 SP}SAVE,{1 SP}F3{1 SP}-
   {1 SP}REDO,{1 SP}F5{1 SP}-{1 SP}EXIT"
62 PRINT LEFT$(BLK$,39);:RETURN
70 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";
   LEFT$(BLK$,37):PRINT"{RA}";LEFT$(BLK$,37):RETURN
75 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";LEFT$(BLK$,37
   ):RETURN
77 REM
78 REM   PARSE ANSWER
79 REM
80 WD$="":WRDS=0:FOR I=1 TO LEN(ANS$)
82 IF ASC(MID$(ANS$,I,1))=32 AND WD$="" THEN 90
84 IF ASC(MID$(ANS$,I,1))=32 AND MID$(ANS$,I+1,1)="
   {1 SP}" THEN 90
86 IF ASC(MID$(ANS$,I,1))=32 OR I=LEN(ANS$) THEN WRDS
   =WRDS+1
87 IF WRDS>2 THEN ERR=TRUE:RETURN
88 WD$=WD$+MID$(ANS$,I,1)
90 NEXT I:FOR I=1 TO LEN(WD$):IF ASC(MID$(WD$,I,1))=32
    AND I<>1 THEN 93
92 NEXT I:RETURN
93 A$=MID$(WD$,1,I):T$=WD$
94 IF A$="THE{1 SP}" OR A$="AN{1 SP}" OR A$="A{1 SP}"
   THEN T$=MID$(WD$,I+1,LEN(T$)-I)
96 ANS$=T$:RETURN
97 REM
98 REM   GET CHARACTERS ROUTINE
99 REM
100 WR=FALSE:RN=1:RW=SP:ANS$="":LOC=0:POKESP+54272,0:
   POKESP,64:POKE SP+1,32:LL=0
102 IF PEEK(198)=0 THEN 102
103 CH=PEEK(631):POKE 198,0
104 IF CH<>13 AND(CH<20 OR (CH>20 AND CH<32) OR CH=44)
    THEN GOSUB 22:GOTO 102
105 IF CH=133 AND SV THEN FLG=2:POKE RW+LOC,32:RETURN
106 IF CH=133 THEN 102
108 IF CH=134 THEN FLG=1:RETURN
```

```
110 IF CH=135 AND EX THEN FLG=3:RETURN
112 IF CH=135 THEN 102
114 IF CH=13 AND ANS$<>LEFT$(BLK$,LEN(ANS$)) THEN
   POKE RW+LOC,32:RETURN
116 IF CH=13 THEN POKE RW+LOC,32:GOTO 100
118 IF CH=20 OR CH=157 THEN 140
120 ANS$=ANS$+CHR$(CH):IF CH>31 AND CH<64 THEN 126
122 IF CH>95 AND CH<128 THEN CH=CH-32:GOTO 126
124 CH=CH-64
126 POKE RW+LOC,CH:LOC=LOC+1:POKE RW+LOC+54272,0:POKER
   W+LOC,64:IFLOC<37THEN 102
127 POKE LOC+RW,32:RN=RN+1:IF RN>NR THEN RN=NR:GOTO 16
   4
128 IF PEEK((LOC-1)+RW)=32 THEN WR=FALSE:GOTO 138
130 T$="":FOR I=37 TO 1 STEP -1:IF ASC(MID$(ANS$,I,1))
   =32 THEN 134
132 T$=MID$(ANS$,I,1)+T$:NEXT:RW=RW+80:LOC=0:POKE RW+5
   4272,0:POKE RW,64:GOTO 102
134 ANS$=MID$(MID$(ANS$,1,I)+BLK$,1,37):PRINT"
   {CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";ANS$:
   PRINT"{RA}";T$
135 WR=TRUE:LL=LEN(ANS$)-LEN(T$)
136 ANS$=ANS$+T$:RW=RW+80:LOC=LEN(T$):POKE RW+LOC+5427
   2,0:POKE RW+LOC,64
137 GOTO 102
138 RW=RW+80:POKE RW+54272,0:POKE RW,64:LOC=0:GOTO 102
140 LOC=LOC-1:IF (LOC<=0) AND (RN=1) THEN 100
141 IF LOC>=0 THEN POKE RW+LOC,64
142 POKE RW+LOC+1,32:IF RN=1 THEN 160
144 IF RN=2 AND LOC<0 THEN RN=1:LOC=36:RW=RW-80:POKE R
   W+LOC,64
145 IF RN=2 AND LOC<0 THEN POKE RW+80,1:GOTO 160
146 IF NOT WR THEN 160
148 IF LL+(LEN(ANS$)-37)>=38 THEN 160
150 T$=LEFT$(ANS$,LL):T1$=MID$(ANS$,38,LEN(ANS$)-37)
152 ANS$=T$+T1$:POKE RW+LOC+1,32:LOC=LEN(ANS$)-1:RN=1:
   RW=RW-80
153 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";
   LEFT$(ANS$,LEN(ANS$)-1):PRINT"{RA}";LEFT$(BLK$,37)
154 POKE RW+LOC,64
160 ANS$=LEFT$(ANS$,LEN(ANS$)-1):GOTO 102
164 POKE RW+54309,0:POKE RW+37,64
166 IF PEEK(198)=0 THEN 166
167 CH=PEEK(631):POKE 198,0
168 IF CH=133 AND SV THEN FLG=2:RETURN
170 IF CH=133 THEN 166
172 IF CH=134 THEN FLG=1:RETURN
174 IF CH=135 AND EX THEN FLG=3:RETURN
176 IF CH=135 THEN 166
178 IF CH=13 AND ANS$<>LEFT$(BLK$,LEN(ANS$)) THEN
   POKE RW+LOC,32:RETURN
180 IF CH=13 THEN 166
182 IF CH=20 OR CH=157 THEN 140
184 GOTO 166
186 WAIT 197,64,64:GET K$:IF K$<>"Y" AND K$<>"N" THEN
   186
187 IF K$="Y" THEN YF=TRUE
188 RETURN
```

```
190 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}";LEFT$(BLK$,38)
192 PRINT"{6 SP}PRESS{1 SP}<RETURN>{1 SP}WHEN{1 SP}DON
    E!{7 SP}":PRINTLEFT$(BLK$,38);
193 GOSUB 22:RETURN
194 Q$=MID$(Q$+BLK$,1,74):A$=MID$(A$+BLK$,1,37):QA$(TN
    )=Q$+A$:RETURN
197 REM
198 REM     SCREEN FORM
199 REM
200 PRINT"{SHIFT/CLR-HOME}{DA}{CTRL/BLK}{6 SP}";TT$:
    PRINT"{CTRL/BLK}{CTRL/RVS ON}{DA}{40 SP}";
205 FOR I=1 TO 17:PRINT"{CTRL/RVS ON}{CTRL/BLK}{1 SP}"
    ;SPC(38);"{1 SP}";:NEXT I:PRINT "{22 SP}";
210 PRINT"{18 SP}":PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}
    {DA}{RA}QUESTION{1 SP}#":PRINT"{COMM/WHT}{DA}{DA}
    {RA}";
215 FORI=0TO36:POKE55617+I,0:POKE55697+I,0:POKE 1345+I
    ,120:POKE 1425+I,120:NEXT
220 PRINT"{DA}{DA}{DA}{DA}{DA}{CTRL/BLK}ANSWER{1 SP}FO
    R{1 SP}QUESTION{1 SP}#"
222 FOR I=0 TO 36:POKE 55977+I,0:POKE 1705+I,120:NEXT
230 PRINT"{DA}{DA}{DA}{RA}{CTRL/BLK}{9 SP}(USE{1 SP}ON
    LY{1 SP}2{1 SP}WORDS)":RETURN
240 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{CTRL/BLK}{3 SP}F1{1 SP}-{1 SP}REDO,{3 SP}F
    3{1 SP}-{1 SP}NEXT,{1 SP}F5{1 SP}-{1 SP}PREV"
245 PRINT "{3 SP}F2{1 SP}-{1 SP}DELETE,{1 SP}F4{1 SP}-
    {1 SP}EXIT{13 SP}":RETURN
250 POKE 55508,0:POKE 55879,0:POKE55509,0:POKE 55510,0
    :POKE 55880,0:POKE 55881,0
252 RN$=RIGHT$(BLK$+STR$(NM),3):FOR I=1 TO 3:POKE 1235
    +I,ASC(MID$(RN$,I,1))
254 POKE 1606+I,ASC(MID$(RN$,I,1))
256 NEXT I:RETURN
280 FOR I=1 TO D:NEXT:RETURN
284 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {RA}{10 SP}RE-ENTER{1 SP}ANSWER!":D=600:GOSUB 280
286 GOSUB 22:PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{RA}";LEFT$(BLK$,38):RETURN
287 REM
288 REM    REDO ROUTINE
289 REM
290 Q$=MID$(QA$(TN),1,74):A$=MID$(QA$(TN),75,37)
300 BOTH=FALSE:PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{2 SP}REDO:";LEFT$(BLK$,30)
310 PRINT"{7 SP}Q{1 SP}-{1 SP}QUESTION,{1 SP}A{1 SP}-
    {1 SP}ANSWER{6 SP}":PRINT"{7 SP}B{1 SP}-{1 SP}BOTH,
    {1 SP}";
312 PRINT"X{1 SP}-{1 SP}EXIT{10 SP}";
314 IF PEEK(198)=0 THEN 314
316 CH=PEEK(631):POKE 198,0
320 IF CH<>81 AND CH<>65 AND CH<>66 AND CH<>88 THEN 31
  4
```

```
322 IF CH=88 THEN RETURN
325 IF CH<>88 THEN GOSUB 190:CNG=TRUE
328 IF CH=81 THEN GOSUB 70:GOTO 334
330 IF CH=65 THEN GOSUB 75:GOTO 340
332 IF CH=66 THEN BOTH=TRUE:GOSUB 70:GOSUB 75
334 SV=FALSE:EX=FALSE:FLG=0:NR=2:SP=1305:GOSUB 100
336 IF FLG=1 THEN GOSUB 70:GOTO 334
337 IF ANS$=LEFT$(BLK$,LEN(ANS$)) THEN 334
338 Q$=ANS$:IF NOT BOTH THEN GOSUB 194:GOTO 290
340 SV=FALSE:EX=FALSE:NR=1:FLG=0:SP=1665:GOSUB 100
342 IF FLG=1 THEN GOSUB 75:GOTO 340
344 ERR=FALSE:GOSUB 80:IF ERR THEN GOSUB 284:GOSUB 75:
   GOTO 340
346 IF ANS$=LEFT$(BLK$,LEN(ANS$)) THEN 340
350 A$=ANS$:GOSUB 194:GOTO 290
397 REM
398 REM    ADDING QUESTIONS
399 REM
400 PRINT"{SHIFT/CLR-HOME}":DE=FALSE
402 TT$="**{2 SP}ADDING{1 SP}QUESTIONS{2 SP}**":GOSUB
   200:GOSUB 60
405 TN=TN+1:IF TN <= 200 THEN 415
408 TN=200:PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}
   {CTRL/BLK}{4 SP}MEMORY{1 SP}IS{1 SP}FULL!!!":GOSUB
   22:GOSUB 50:RETURN
415 NM=TN:GOSUB 250:GOSUB 70:GOSUB 75
420 NR=2:FLG=0:SV=TRUE:EX=TRUE:SP=1305:GOSUB 100
422 IF FLG=1 THEN GOSUB 70:GOTO 420
424 IF FLG=3 THEN TN=TN-1:RETURN
426 IF ANS$=LEFT$(BLK$,LEN(ANS$)) THEN 420
428 Q$=ANS$
430 NR=1:FLG=0:SV=TRUE:EX=TRUE:SP=1665:GOSUB 100
432 IF FLG=1 THEN GOSUB 70:GOSUB 75:GOTO 420
434 IF FLG=3 THEN TN=TN-1:RETURN
435 IF ANS$=LEFT$(BLK$,LEN(ANS$)) THEN 430
436 ERR=FALSE:GOSUB 80:IF ERR THEN  GOSUB 284:GOSUB 75
   :GOTO 430
437 A$=ANS$
438 IF FLG=2 THEN GOSUB 194:GOSUB 22:GOSUB 22:CNG=TRUE
   :POKE 1981,32:GOTO 405
439 GOSUB 22:POKE 56253,0:POKE 1981,64
440 IF PEEK(198)=0 THEN 440
441 CH=PEEK(631):POKE 198,0
442 IF CH<133 OR CH>135 THEN 440
444 IF CH=133 THEN FLG=2:GOTO 438
446 IF CH=135 THEN TN=TN-1:RETURN
448 GOSUB 194:GOSUB 290:GOSUB 60:GOSUB 22:GOTO 440
497 REM
498 REM    DISPLAY/EDIT QUESTIONS
499 REM
500 IF TN>0 THEN 510
502 PRINT"{SHIFT/CLR-HOME}{CTRL/BLK}{DA}{DA}{DA}
   {3 SP}THERE{1 SP}ARE{1 SP}NO{1 SP}QUESTIONS{1 SP}AN
   D{1 SP}ANSWERS":PRINT"{DA}{3 SP}IN{1 SP}MEMORY
   {1 SP}TO{1 SP}";
504 PRINT"DISPLAY/EDIT.":GOSUB 22:GOSUB 50:RETURN
510 ARP=1:DE=TRUE
520 TT$="**{2 SP}DISPLAY/EDIT{1 SP}QUESTIONS{2 SP}**":
   GOSUB 200
530 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";
   MID$(QA$(ARP),1,37):PRINT"{RA}";MID$(QA$(ARP),38,37
   )
```

```
535 PRINT:PRINT"{DA}{DA}{DA}{RA}";MID$(QA$(ARP),75,37)
540 NM=ARP:GOSUB 250:GOSUB 240:POKE 56241,0:POKE 1971,
   64:POKE 649,1
542 IF PEEK(198)=0 THEN 542
543 CH=PEEK(631):POKE 198,0
544 IF CH=133 THEN TT=TN:TN=ARP:GOSUB 290:GOSUB 194:TN
   =TT:GOTO 520
546 IF CH=134 THEN ARP=ARP+1:GOTO 558
548 IF CH=135 THEN ARP=ARP-1:GOTO 566
550 IF CH=137 THEN 576
552 IF CH=138 THEN RETURN
553 GOTO 542
555 REM
556 REM  GET NEXT RECORD
557 REM
558 IF ARP<=TN THEN GOSUB 250:GOTO 530
560 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{3 SP}THERE{1 SP}ARE{1 SP}NO{1 SP}MORE
   {1 SP}QUESTIONS!{8 SP}"
562 PRINTLEFT$(BLK$,38);:GOSUB 22:D=600:GOSUB 280:ARP
   =ARP-1:GOTO 540
563 REM
564 REM  GET PREVIOUS RECORD
565 REM
566 IF ARP>=1 THEN GOSUB 250:GOTO 530
568 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{3 SP}CAN'T{1 SP}GO{1 SP}BELOW{1 SP}RECORD
   {1 SP}#1!{11 SP}"
570 PRINTLEFT$(BLK$,38);:GOSUB 22:D=600:GOSUB 280:ARP
   =ARP+1:GOTO 540
573 REM
574 REM  DELETE RECORD?
575 REM
576 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}";LEFT$(BLK$,39)
578 PRINT"{5 SP}DELETE{1 SP}THIS{1 SP}RECORD?{1 SP}(Y/
   N)":YF=FALSE:GOSUB 22
579 GOSUB 186:IF NOT YF THEN 540
580 IF TN-1=0 THEN QA$(ARP)="":TN=0:GOTO 500
582 IF ARP=TN THEN QA$(TN)="":ARP=ARP-1:GOTO 586
584 QA$(ARP)=QA$(TN)
586 TN=TN-1:CNG=TRUE:IF TN<1 THEN 500
588 NM=ARP:GOSUB 250:GOTO 530
597 REM
598 REM   MAIN MENU SELECTION 3 & 4
599 REM
600 IF NOT CNG AND SL=4 THEN POKE 53281,6:POKE 53280,1
   4:PRINT"{COMM/BLU}{SHIFT/CLR-HOME}":END
610 IF NOT CNG THEN 810
612 NUMQ=TN:YF=FALSE:PRINT"{SHIFT/CLR-HOME}{DA}{DA}
   {DA}{DA}{DA}{5 SP}SAVE{1 SP}CHANGES?{1 SP}(Y/N)":
   GOSUB 22
620 GOSUB 186:IF NOT YF AND SL=4 THEN POKE 53281,6:
   POKE 53280,14:PRINT"{COMM/BLU}{SHIFT/CLR-HOME}":
   END
630 IF NOT YF THEN 810
```

```
632 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{4 SP}PLEASE
   {1 SP}INSERT{1 SP}A{1 SP}CASSETTE{1 SP}WHERE"
633 PRINT"{4 SP}YOU{1 SP}WANT{1 SP}YOUR{1 SP}NEW
   {1 SP}QUESTIONS{1 SP}AND"
634 PRINT"{4 SP}ANSWERS{1 SP}TO{1 SP}BE{1 SP}SAVED.
   {DA}{DA}"
636 PRINT"{4 SP}THE{1 SP}SCREEN{1 SP}WILL{1 SP}GO
   {1 SP}BLANK.{2 SP}BUT":PRINT"{4 SP}DON'T{1 SP}PANIC
   !";
637 PRINT"{2 SP}THE{1 SP}SCREEN{1 SP}WILL":PRINT"
   {4 SP}RETURN{1 SP}AS{1 SP}SOON{1 SP}AS{1 SP}THE
   {1 SP}COMMODORE"
638 PRINT"{4 SP}HAS{1 SP}FINISHED{1 SP}SAVING{1 SP}THE
   {1 SP}FILE.":GOSUB 50:PRINT"{SHIFT/CLR-HOME}"
640 OPEN1,1,1,"TRIVIA{1 SP}FILE"
650 PRINT#1,NUMQ
655 FOR I=1 TO NUMQ
657 PRINT#1,MID$(QA$(I),1,74)
658 PRINT#1,MID$(QA$(I),75,37)
659 NEXT I:CLOSE 1
660 IF SL=4 THEN POKE 53280,14:POKE 53281,6:PRINT"
   {COMM/BLU}{SHIFT/CLR-HOME}":END
795 REM
800 REM  FIRST MENU
805 REM
810 CNG=FALSE:NUMQ=0:PRINT"{SHIFT/CLR-HOME}{DA}{DA}
   {CTRL/BLK}{8 SP}**{2 SP}TRIVIA{1 SP}DATA{1 SP}BASE
   {2 SP}**"
820 PRINT"{DA}{DA}{DA}{2 SP}DO{1 SP}YOU{1 SP}WANT
   {1 SP}TO:{DA}{DA}":PRINT "{4 SP}1.{1 SP}CREATE
   {1 SP}A{1 SP}NEW{1 SP}TRIVIA{1 SP}FILE"
830 PRINT "{DA}{4 SP}2.{1 SP}LOAD{1 SP}AN{1 SP}EXISTIN
   G{1 SP}FILE":PRINT"{DA}{4 SP}3.{1 SP}EXIT{1 SP}DATA
   {1 SP}BASE{1 SP}PROGRAM"
840 PRINT"{DA}{DA}{4 SP}PLEASE{1 SP}SELECT{1 SP}(1-3)"
   :GOSUB 22
850 WAIT 197,64,64:GET SL$:IF SL$<"1" OR SL$>"3" THEN
   850
860 POKE 1688,ASC(SL$):IF VAL(SL$)=3 THENPOKE 53281,6:
   POKE53280,14:PRINT"{COMM/BLU}{SHIFT/CLR-HOME}":END
870 IF VAL(SL$)=1 THEN 1000
880 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{3 SP}PLEASE
   {1 SP}INSERT{1 SP}THE{1 SP}CASSETTE"
890 PRINT"{DA}{3 SP}WHERE{1 SP}YOUR{1 SP}QUESTIONS
   {1 SP}AND{1 SP}ANSWERS":PRINT"{DA}{3 SP}HAVE{1 SP}B
   EEN{1 SP}SAVED."
900 PRINT"{DA}{DA}{3 SP}AS{1 SP}SOON{1 SP}AS{1 SP}YOU
   {1 SP}PRESS{1 SP}'PLAY'":PRINT"{DA}{3 SP}THE{1 SP}S
   CREEN{1 SP}WILL{1 SP}GO{1 SP}BLANK."
910 PRINT "{DA}{3 SP}DON'T{1 SP}PANIC!{3 SP}ONCE
   {1 SP}THE{1 SP}FILE{1 SP}HAS{1 SP}"
920 PRINT"{DA}{3 SP}BEEN{1 SP}LOADED{1 SP}THE{1 SP}SCR
   EEN{1 SP}WILL{1 SP}RETURN":PRINT"{DA}{3 SP}TO
   {1 SP}ITS{1 SP}NORMAL{1 SP}COLOR.
930 GOSUB 22:GOSUB 50:PRINT"{SHIFT/CLR-HOME}":OPEN1,1,
   0,"TRIVIA{1 SP}FILE"
940 INPUT# 1,NUMQ:FOR I=1 TO NUMQ
945 INPUT# 1,Q$
946 INPUT# 1,A$:QA$(I)=Q$+A$
950 NEXT I:CLOSE 1
997 REM
```

```
998 REM     DB MAIN MENU
999 REM
1000 TN=NUMQ
1005 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{CTRL/BLK}{7 SP}**
   {2 SP}TRIVIA{1 SP}DB{1 SP}MAIN{1 SP}MENU{2 SP}**"
1010 PRINT"{DA}{DA}{DA}{7 SP}1.{1 SP}ADD{1 SP}QUESTION
   S":PRINT"{DA}{7 SP}2.{1 SP}DISPLAY/EDIT{1 SP}QUESTI
   ONS"
1020 PRINT"{DA}{7 SP}3.{1 SP}CHANGE{1 SP}FILES":PRINT"
   {DA}{7 SP}4.{1 SP}EXIT{1 SP}PROGRAM"
1030 PRINT "{DA}{DA}{7 SP}PLEASE{1 SP}SELECT{1 SP}(1-4
   )":GOSUB 22
1040 WAIT 197,64,64:GET SL$:IF SL$<"1" OR SL$>"4"
   THEN 1040
1045 POKE 55923,0:POKE 1651,ASC(SL$):SL=VAL(SL$):ON SL
    GOSUB 400,500,600,600
1050 GOTO 1005
```

VARIABLE LISTING

The following is a listing of the main variables in the cassette data base. It includes the most important variables and may not include all the variables in the program.

A$	Temporarily holds the answer before packing and concatenating to the question.
ANS$	Holds the answer. Maximum length is 37 characters.
ARP	Holds the current location of the array QA$ for displaying the record to the screen. Used in display/edit menu.
BLK$	Blank string the size of the maximum size field (74). Used for packing the question and answer to their maximum length.
BOTH	Boolean flag set to indicate if the user wants to REDO both the question and the answer. If true then the user wants to edit both parts of the record.
CH	Holds the ASCII value of the character that was pressed. The ASCII value is taken from the buffer location at 631.

CNG	Boolean flag. Indicates if any change has been made to the existing file in memory.
DE	Flag set to indicate whether the program execution came from the *D*isplay/*E*dit menu. If true, it did. Makes sure you return to display/edit menu.
ERR	Flag used to indicate if an error occurred within a procedure. This type of an error is not a system error but a user error (invalid response, or illegal characters, etc.).
EX	Boolean flag. Indicates whether or not to allow the user to use the **F5** or **F4** key.
FALSE	Boolean flag set to Ø. Indicates a false value.
FLG	Flag set to indicate if one of the command keys (**F1**, **F3**, etc.) has been pressed. Returns a 1 if **F3** was pressed, a 2 if **F1** was pressed and a 3 if **F5** was pressed.
I,D,L	Looping variables for FOR . . . NEXT statements.
K$	Used to get a key from the keyboard via GET statement.
LL	Length of the first line of the question. Used to determine when the question needs to backwrap around to the beginning line.
LOC	Location of the cursor on the current row.
M	Sound chip beginning address location.
NM	Record number to print. Same as the question number. Shows what question the user is currently entering.
NR	The number of rows the cursor is allowed to move. For the question NR is set to 2; for the answer it is set to 1.
NUMQ	Number of questions currently loaded into memory or set to Ø initially for creating a new file.

QA$(2ØØ)	Holds the questions and answers in a concatenated string. Total string length is 111 characters—74 for the question and 37 for the answer.
Q$	Holds the question. Maximum length is 74 characters.
RN	Counter to keep track of how many rows the cursor has moved compared to how many it is allowed to move.
RT$	Used in GET statement to indicate when a <RETURN> key is needed.
RW	Is used for keeping the cursor on the same line until end of line is reached, then to move it down to the next line.
SL	Holds the selection number the user chose from the main menu.
SL$	Holds the string representation of the number pressed from one of the menus.
SP	Starting row location to place the cursor and where to start printing characters as they are typed.
SV	Boolean flag. Indicates whether or not to allow the user to press **F1** while typing the question or answer.
T$	Temporarily holds the answer after it is converted to uppercase letters.
T1$	Temporarily holds the wraparound answer to the question.
TN	Holds the array number of the next location to store the question and answer.
TRUE	Boolean flag indicates that the result is indeed true.
TT	Temporarily holds the total number of questions and answers in memory.

TT$	Holds the title string to be printed at the top of each screen.
WD$	Temporarily holds the new answer after parsing.
WR	Holds a true if the question wraps around to the next line, or a false if the question didn't wrap around.
WRDS	Counter used to count how many words are in answer.
YF	Boolean flag to indicate whether the user responded with a yes (Y) or a no (N).

Appendix B

The Cassette Game

PROGRAM LISTING

```
5 REM
6 REM  CASSETTE VERSION - GAME
7 REM
10 DIM QA$(200),QNU(30),CK(30),ND(8),NT(4):POKE 53280,
   13:POKE 53281,13
11 BLK$="{37 SP}":FALSE=0:TRUE=NOT FALSE
12 M=54272:FOR L=MTO M+24:POKE L,0:NEXT:GOTO 800
15 RN$=RIGHT$(BLK$+STR$(TM),2):POKE 55487,0:POKE 55488
   ,0
16 POKE 1215,ASC(MID$(RN$,1,1)):POKE 1216,ASC(MID$(RN$
   ,2,1)):RETURN
22 M=54272:POKE M+24,15:POKEM+5,9:POKEM+6,0:POKE M+1,2
   5:POKE M,30
24 POKE M+4,33:FOR I=1 TO 150:NEXT:POKE M+4,32:POKE M
   +24,0:RETURN
30 M=54272:POKE M+24,15:POKEM+5,9:POKEM+6,0:POKE M+1,6
   :POKE M,71
32 POKE M+4,33:FOR I=1 TO 150:NEXT:POKE M+4,32:POKE M
   +24,0:RETURN
47 REM
48 REM   WAITS FOR THE RETURN KEY
49 REM
50 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{5 SP}PRESS{2 SP}<RETURN>{2 SP}TO
   {1 SP}CONTINUE";
55 WAIT 197,64,64:GET RT$:IF RT$<>CHR$(13) THEN 55
58 RETURN
60 FOR I=1 TO D:NEXT:RETURN
61 REM
62 REM  DISPLAY STATUS OF PLAYER
63 REM
64 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}{RA}
   {RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}";PL;"{RA}{RA}{RA}{RA}
   {RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}";LEFT$(PN$(PL)
   ,10)
65 FOR Q=0 TO 3:POKE 1832+Q,32:POKE 1849+Q,32:NEXT:
   POKE 1853,32:POKE 1854,32
66 POKE 1855,32:POKE 1856,32:POKE 1857,32:POKE 1858,32
68 PRINT"{DA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}";SC(PL);"
   {RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}
   {RA}{RA}{RA}";NC(PL);"/";NQ(PL):RETURN
72 REM
73 REM   ERASES ANSWER
74 REM
75 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{RA}";BLK$:RETURN
77 REM
78 REM  PARSING ROUTINE
79 REM
80 WD$="":WRDS=0:FOR I=1 TO LEN(ANS$)
82 IF ASC(MID$(ANS$,I,1))=32 AND WD$="" THEN 90
84 IF ASC(MID$(ANS$,I,1))=32 AND MID$(ANS$,I+1,1)="
   {1 SP}" THEN 90
86 IF ASC(MID$(ANS$,I,1))=32 OR I=LEN(ANS$) THEN WRDS
   =WRDS+1
87 IF WRDS>2 THEN ERR=TRUE:RETURN
```

```
88 WD$=WD$+MID$(ANS$,I,1)
90 NEXT'I:FOR I=1 TO LEN(WD$):IF ASC(MID$(WD$,I,1))=32
    AND I<>1 THEN 93
92 NEXT I:RETURN
93 A$=MID$(WD$,1,I):T$=WD$
94 IF A$="THE{1 SP}" OR A$="AN{1 SP}" OR A$="A{1 SP}"
   THEN T$=MID$(WD$,I+1,LEN(T$)-I)
96 ANS$=T$:RETURN
97 REM
98 REM   GET ANSWER ROUTINE
99 REM
100 POKE 198,0:POKE 649,1:TI$="000000"
101 RW=SP:ANS$="":LOC=0:POKESP+54272,0:POKESP,64:POKES
   P+1,32
102 IFPEEK(198)=0THENTM=VAL(RIGHT$(TI$,2)):TM=TP-TM:
   IFTM<0 THEN TU=TRUE:RETURN
103 GOSUB 15:IF PEEK(198)=0 THEN 102
104 CH=PEEK(631):POKE 198,0
106 IF CH<>13 AND (CH<20 OR (CH>20 AND CH<32)) THEN 10
   2
109 IF CH=133 THEN EX=TRUE:RETURN
112 IF CH=13 AND ANS$<>"" THEN POKE RW+LOC,32:RETURN
113 IF CH=13 THEN 102
114 IF CH=20 OR CH=157 THEN 138
115 TM=VAL(RIGHT$(TI$,2)):TM=TP-TM:IF TM<0 THEN TU=TRU
   E:RETURN
116 GOSUB 15
117 ANS$=ANS$+CHR$(CH):IF CH>31 AND CH<64 THEN 122
118 IF CH>95 AND CH<128 THEN CH=CH-32:GOTO 122
120 CH=CH-64
122 POKE RW+LOC,CH:LOC=LOC+1:POKE RW+LOC+54272,0:POKER
   W+LOC,64:IFLOC<37THEN 102
124 POKE LOC+RW,32:POKE RW+54309,0:POKE RW+37,64
130 IF PEEK(198)=0 THENTM=VAL(RIGHT$(TI$,2)):TM=TP-TM:
   IFTM<0THENTU=TRUE:RETURN
131 GOSUB 15:IF PEEK(198)=0 THEN 130
132 IF CH=13 THEN RETURN
134 IF CH=20 OR CH=157 THEN 138
135 IF CH=133 THEN EX=TRUE:RETURN
136 GOTO 130
138 LOC=LOC-1:IF LOC<1 THEN 101
140 POKE RW+LOC,64:POKE RW+LOC+1,32
141 TM=VAL(RIGHT$(TI$,2)):TM=TP-TM:IF TM<0 THEN TU=TRU
   E:RETURN
142 GOSUB 15
144 ANS$=LEFT$(ANS$,LEN(ANS$)-1):GOTO 102
183 REM
184 REM  Y/N ROUTINE
185 REM
186 WAIT 197,64,64:GET K$:IF K$<>"Y" AND K$<>"N" THEN
   186
187 IF K$="Y" THEN YF=TRUE
188 RETURN
197 REM
198 REM   GAME PLAYING
199 REM
200 GOSUB 64:NM=AL:GOSUB 450:TM=25:GOSUB 15:TP=25
202 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";
   LEFT$(QA$(QNU(AL)),37):PRINT"{RA}";MID$(QA$(QNU(AL)
   ),38,37)
203 PS=PL
```

```
204 PD=FALSE:FOR I=1 TO DB:IF ND(I)=QNU(AL) THEN PD=TR
   UE:GOTO 210
206 NEXT:PD=FALSE:FOR I=1 TO T3:IF NT(I)=QNU(AL) THEN
   PT=TRUE:GOTO 210
208 NEXT I:PT=FALSE
210 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{RA}";LEFT$(BLK$,37):GOSUB 75
212 IF PD THEN PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{RA}{10 SP}DOUBLE{1 SP}VALUE!":
   GOSUB 22:GOSUB 22
214 IF PT THEN PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{RA}{10 SP}TRIPLE{1 SP}VALUE!":
   GOSUB 22:GOSUB 22
220 TU=FALSE:EX=FALSE:SP=1505:GOSUB 100:PI=TM
222 IF EX THEN 600
224 IF TU THEN 240
226 IF PD THEN PI=PI*2
228 IF PT THEN PI=PI*3
230 GOSUB 80:ANS$=MID$(ANS$+BLK$,1,37):CA$=MID$(QA$(QN
   U(AL)),75,37)
232 IF CA$=ANS$ THEN 250
240 GOSUB 30:GOSUB 30:D=300:GOSUB 60:NQ(PL)=NQ(PL)+1:P
   L=PL+1:IF PL>NP THEN PL=1
242 IF PL<>PS THEN TP=12:GOSUB 15:GOSUB 64:GOSUB 75:
   GOTO 220
244 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{RA}THE{1 SP}CORRECT{1 SP}ANSWER{1 SP}IS:":
   GOSUB 30
245 PRINT"{DA}{RA}";MID$(QA$(QNU(AL)),75,37):D=1000:
   GOSUB 60
246 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{RA}ANSWER{19 SP}"
248 GOSUB 75:GOTO 260
250 GOSUB 22:GOSUB 22:SC(PL)=SC(PL)+PI:NC(PL)=NC(PL)+1
   :NQ(PL)=NQ(PL)+1
252 GOSUB 64:D=500:GOSUB 75:GOSUB 60
260 AL=AL+1:IF AL>MN THEN 280
262 PL=PS+1:IF PL>NP THEN PL=1
264 PD=FALSE:PT=FALSE:GOTO 200
280 D=400:GOSUB 64:GOSUB 60:GOTO 750
447 REM
448 REM  DISPLAY THE QUESTION NUMBER
449 REM
450 POKE 55468,0:POKE 55469,0
452 RN$=RIGHT$(BLK$+STR$(NM),2):POKE 1196,ASC(MID$(RN$
   ,1,1))
456 POKE 1197,ASC(MID$(RN$,2,1)):RETURN
497 REM
498 REM   CHECKS FOR DUPLICATE NUMBERS
499 REM
500 IF X=0 THEN ERR=FALSE:RETURN
502 FOR Q=1 TO X:IF WN=CK(Q) THEN ERR=TRUE:RETURN
504 NEXT:ERR=FALSE:RETURN
597 REM
598 REM   QUIT GAME EARLY?
599 REM
600 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{RA}{9 SP}ARE{1 SP}YOU{1 SP}SURE?{1 SP}(Y/N
   )":GOSUB 22
```

```
610 YF=FALSE:GOSUB 186:IF NOT YF THEN PRINT"
    {CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {RA}";
620 IF NOT YF THEN PRINT"{13 SP}F1{1 SP}-{1 SP}QUIT
    {7 SP}":TP=PI:GOSUB 75:GOTO 220
747 REM
748 REM  END OF GAME STATUS
749 REM
750 GOSUB 22:PRINT"{SHIFT/CLR-HOME}{DA}{DA}{CTRL/BLK}
    {9 SP}**{1 SP}FINAL{1 SP}STATS{1 SP}**":PRINT"{DA}
    {DA}{DA}{3 SP}NAME{8 SP}";
752 PRINT"{1 SP}SCORE{9 SP}NO/NO":PRINT"{CTRL/RVS ON}
    {39 SP}"
758 FOR I=1 TO NP:PN(I)=I:NEXT:FOR J=1 TO NP-1:J1=J
760 FOR K=J+1 TO NP:IF SC(J1)<SC(K) THEN J1=K
762 NEXT K:IF J1<>J THEN T=SC(J):SC(J)=SC(J1):SC(J1)=T
    :T=PN(J):PN(J)=PN(J1)
764 IF J1<>J THEN PN(J1)=T
766 NEXT J
768 FOR I=1 TO NP:PRINTLEFT$(PN$(PN(I)),10);"{7 SP}";S
    C(I);"{6 SP}";
769 PRINT NO(PN(I));"/";NQ(PN(I)):NEXT
770 IF EX THEN D=1000:GOSUB 60:GOTO 776
772 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}ANOTHER{1 SP}GAME?{1 SP}(Y/N)":GOSUB 22:YF=FALS
    E
774 GOSUB 186:IF YF THEN X=RND(-1):GOTO 885
776 POKE 53280,14:POKE 53281,6:PRINT"{SHIFT/CLR-HOME}
    {COMM/BLU}{DA}{DA}{DA}HAVE{1 SP}A{1 SP}NICE{1 SP}DA
    Y!":END
797 REM
798 REM    INTRODUCTION SCREEN
799 REM
800 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}
    {CTRL/BLK}{3 SP}WELCOME{1 SP}TO{1 SP}COMMODORE
    {1 SP}64{1 SP}TRIVIA{1 SP}GAME"
810 PRINT"{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{6 SP}NEED{1 SP}INSTRUCTION
    S?{1 SP}(Y/N)":YF=FALSE:GOSUB 22
820 GOSUB 22:GOSUB 186:IF NOT YF THEN 860
822 PRINT"{SHIFT/CLR-HOME}{CTRL/BLK}{9 SP}INSTRUCTIONS
    "
824 PRINT"{DA}{2 SP}THE{1 SP}OBJECT{1 SP}OF{1 SP}THE
    {1 SP}GAME{1 SP}IS{1 SP}TO{1 SP}GAIN{1 SP}AS":
    PRINT"MANY{1 SP}POINTS{1 SP}AS";
826 PRINT"{1 SP}POSSIBLE{1 SP}BY{1 SP}ANSWERING":
    PRINT"RANDOM{1 SP}TRIVIA{1 SP}QUESTIONS."
828 PRINT"{DA}{2 SP}EACH{1 SP}QUESTION{1 SP}IS{1 SP}WO
    RTH{1 SP}A{1 SP}STARTING":PRINT"VALUE{1 SP}OF
    {1 SP}25{1 SP}POINTS.";
830 PRINT"{2 SP}THE{1 SP}FASTER{1 SP}YOU":PRINT"ANSWER
    {1 SP}THE{1 SP}QUESTION{1 SP}THE{1 SP}MORE{1 SP}POI
    NTS";
832 PRINT"{1 SP}YOU{1 SP}CAN{1 SP}OBTAIN.{2 SP}AS
    {1 SP}TIME{1 SP}GOES{1 SP}BY,{1 SP}THE"
834 PRINT "POINT{1 SP}VALUE{1 SP}DECREASES.":PRINT"
    {DA}{2 SP}IF{1 SP}A{1 SP}PLAYER{1 SP}MISSES{1 SP}TH
    E{1 SP}ANSWER,"
```

```
836 PRINT"THE{1 SP}NEXT{1 SP}PLAYER{1 SP}IN{1 SP}LINE
   {1 SP}GETS{1 SP}A{1 SP}CHANCE":PRINT"TO{1 SP}ANSWER
   {1 SP}THE{1 SP}QUESTION.";
838 PRINT"{2 SP}ONCE{1 SP}ALL{1 SP}THE":PRINT"PLAYERS
   {1 SP}HAVE{1 SP}HAD{1 SP}A{1 SP}CHANCE{1 SP}TO
   {1 SP}ANSWER"
840 PRINT"THE{1 SP}QUESTION,{1 SP}THEN{1 SP}THE{1 SP}C
   ORRECT{1 SP}ANSWER":PRINT"IS{1 SP}DISPLAYED."
842 PRINT"{DA}{2 SP}THE{1 SP}GAME{1 SP}CAN{1 SP}BE
   {1 SP}PLAYED{1 SP}BY{1 SP}1-4":PRINT"PLAYERS.";
844 PRINT"{3 SP}GOOD{1 SP}LUCK!...HAVE{1 SP}FUN!"
846 PRINT"{DA}{CTRL/BLK}*{1 SP}ALL{1 SP}ANSWERS{1 SP}M
   UST{1 SP}MATCH{1 SP}EXACTLY!"
850 GOSUB 50
857 REM
858 REM    LOAD CASSETTE FILE
859 REM
860 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{2 SP}PL
   EASE{1 SP}INSERT{1 SP}THE{1 SP}CASSETTE{1 SP}THAT
   {1 SP}THE":PRINT"{2 SP}QUESTIONS{1 SP}AND";
862 PRINT"{1 SP}ANSWERS{1 SP}ARE{1 SP}STORED{1 SP}ON."
864 PRINT"{DA}{2 SP}IT{1 SP}IS{1 SP}GOING{1 SP}TO
   {1 SP}TAKE{1 SP}A{1 SP}WHILE{1 SP}TO{1 SP}LOAD":
   PRINT"{2 SP}THE{1 SP}FILE.{2 SP}PLEASE{1 SP}BE";
866 PRINT"{1 SP}PATIENT{1 SP}WHILE{1 SP}I":PRINT"
   {2 SP}LOAD{1 SP}THE{1 SP}FILE."
868 PRINT"{DA}{DA}{2 SP}DON'T{1 SP}PANIC!{2 SP}WHEN
   {1 SP}THE{1 SP}SCREEN{1 SP}GOES":PRINT"{2 SP}BLANK.
   {2 SP}IT{1 SP}WILL";
870 PRINT"{2 SP}RETURN{1 SP}AS{1 SP}SOON":PRINT"
   {2 SP}AS{1 SP}THE{1 SP}CASSETTE{1 SP}IS{1 SP}THROUG
   H{1 SP}BEING"
871 PRINT"{2 SP}LOADED."
872 GOSUB 22:GOSUB 50:PRINT"{SHIFT/CLR-HOME}"
874 OPEN1,1,0,"TRIVIA{1 SP}FILE"
876 INPUT# 1,NUMQ:FOR I=1 TO NUMQ
878 INPUT# 1,Q$
880 INPUT# 1,A$:QA$(I)=Q$+A$
881 NEXT I:CLOSE 1
882 REM
883 REM    SELECT QUESTIONS (DBL & TRP POINTS)
884 REM
885 GOSUB 22:PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{3 SP}P
   LEASE{1 SP}WAIT...SELECTING{1 SP}QUESTIONS"
886 IF NUMQ<=30 THEN MN=NUMQ:GOTO 904
888 MN=30:FOR I=1 TO MN
890 WN=INT(RND(0)*NUMQ)+1:X=I-1:ERR=FALSE:GOSUB 500:
   IF ERR THEN 890
900 QNU(I)=WN:CK(I)=WN:NEXT I:GOTO 920
904 FOR I=1 TO MN
906 WN=INT(RND(0)*MN)+1:X=I-1:ERR=FALSE:GOSUB 500:IF E
   RR THEN 890
908 QNU(I)=WN:CK(I)=WN:NEXT I
920 DB=INT(MN*.25)+1:T3=INT(MN*.10)+1:FOR I=1 TO DB
924 WN=INT(RND(0)*MN)+1:X=I-1:ERR=FALSE:GOSUB 500:IF E
   RR THEN 924
926 ND(I)=QNU(WN):CK(I)=QNU(WN):NEXT I:FOR I=1 TO T3:C
   K(I)=0:NEXT:FOR I=1 TO T3
928 WN=INT(RND(0)*MN)+1:X=I-1:ERR=FALSE:GOSUB 500:IF E
   RR THEN 928
929 FOR J=1 TO T3:T(J)=CK(J):NEXT
```

```
930 FOR J=1 TO DB:CK(J)=ND(J):NEXT:ERR=FALSE:X=DB:
   GOSUB 500
932 IF ERR THEN 928
933 FOR J=1 TO T3:CK(J)=T(J):NEXT
934 NT(I)=QNU(WN):CK(I)=QNU(WN):NEXT I
935 REM
936 REM    NUMBER OF PLAYERS AND NAMES
937 REM
940 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{5 SP}NU
   MBER{1 SP}OF{1 SP}PLAYERS{1 SP}(1-4)":GOSUB 22
942 WAIT 197,64,64:GET NP$:IF NP$="" THEN 942
944 IF ASC(NP$)<49 OR ASC(NP$)>52 THEN 942
946 POKE55526,0:POKE 1254,ASC(NP$):NP=ASC(NP$)-48:FOR
   I=1 TO NP
947 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}"
948 PRINT"PLAYER{1 SP}#{1 SP}";I:PN$(I)="":INPUT "
   {DA}{DA}{2 SP}NAME";PN$(I)
950 IF PN$(I)="" THEN 947
952 PN$(I)=MID$(PN$(I)+BLK$,1,10):NEXT I
957 REM
958 REM    DISPLAY GAME SCREEN
959 REM
960 PRINT"{SHIFT/CLR-HOME}{DA}{CTRL/BLK}{10 SP}**
   {1 SP}TRIVIA{1 SP}GAME{1 SP}**"
961 PRINT"{DA}{CTRL/RVS ON}{40 SP}";
962 FOR I=1 TO 13:PRINT"{CTRL/RVS ON}{1 SP}";SPC(38);"
   {1 SP}";:NEXT
964 PRINT"{40 SP}";
966 FOR I=1 TO 5:PRINT"{1 SP}";SPC(38);"{1 SP}";:NEXT:
   PRINT"{20 SP}";
967 PRINT"{20 SP}{CTRL/RVS OFF}";
970 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{RA}QUESTION
   {1 SP}#":PRINT"{DA}{DA}{DA}{DA}{DA}{RA}ANSWER:"
972 FOR I=0 TO 36:POKE 55817+I,0:POKE 1545+I,120:NEXT
974 PRINT"{DA}{DA}{DA}{RA}{9 SP}{USE{1 SP}ONLY{1 SP}2
   {1 SP}WORDS}{CTRL/BLK}"
976 PRINT"{DA}{DA}{DA}{RA}PLAYER{1 SP}#{10 SP}NAME:":
   PRINT"{DA}{RA}SCORE:  {11 SP}NC/NQ:"
978 PRINT"{DA}{RA}{CTRL/BLK}{13 SP}F1{1 SP}-{1 SP}QUIT
   {1 SP}":AL=1:PL=1:FOR I=1 TO NP:SC(I)=0
980 NC(I)=0:NQ(I)=0:NEXT I:GOTO 200
```

VARIABLE LISTING

The following is a listing of the main variables in the cassette game. It includes the most important variables and may not include all the variables in the program.

A$	Holds the first word of the answer to see if it matches "a", "an", or "the". Also holds the answer loaded into memory from tape.
AL	Holds the array number of the question that is displayed.
ANS$	Holds the answer that the player has typed. Maximum length is 37 characters.
BLK$	Blank string the size of the maximum size field (37). Used for packing the answer to its maximum length.
CA$	Holds the correct answer. Used for comparing the two answers for equality.
CH	Holds the ASCII value for one character at a time.
CK()	Holds the number for checking against duplicate numbers when selecting random questions, double and triple point value.
DB	Holds the number of double point questions that this game is supposed to have.
ERR	Flag used to indicate if an error occurred within a procedure.
EX	Holds the boolean flag for quitting the game early.
FALSE	Boolean flag set to Ø. Indicates a false value.
I,J,K,L,D & Q	Looping variables for FOR . . . NEXT statements.
K$	Holds the key pressed using GET statement.
LOC	Location on the answer line of the cursor.
M	Holds the sound chip memory location.
MN	Maximum number of questions to use. 30 if NUMQ>30 or is the same as NUMQ.
NC()	Holds number of correct questions that the player answered.

ND(8)	Holds the randomly generated numbers that are to be assigned double point values.
NM	Holds the question number to be printed to the screen.
NP	The total number of players playing the game (from one to four).
NQ()	Holds the numbers of questions that each player was asked.
NT(4)	Holds the question numbers for the triple point questions.
NUMQ	Number of questions currently loaded into memory from the cassette file.
PD	Boolean flag to indicate if the current question is to be double value or not. Used for printing the DOUBLE VALUE message.
PI	Holds the number of points the player gets when answering the question correctly.
PL	Holds the current player number to display on the screen.
PN$()	Holds the players names. Up to 10 characters long, but only 6 are displayed on the game screen. The whole name is displayed at the end of the game in the statistic screen.
PS	Holds the player number of who started first. This keeps the players going in the same order no matter who answers a missed question.
PT	Holds the triple value flag set for printing or not printing the TRIPLE VALUE message.
Q$	Holds the question loaded into memory from the tape.
QA$(2ØØ)	Holds the questions and answers in a concatenated string. Total string length is 111 characters. 74 for the question and 37 for the answer.

QNU(3Ø)	Holds the randomly generated sequence of numbers in which to display the questions in a random format.
RN$	Holds the string value of the internal timer before printing the clock.
RW	Starting row location to place the cursor and where to start printing characters as they are typed.
SC()	Holds the player's score.
SP	Starting cursor position on the screen for entering the answer.
T$	Holds the word(s) to parse out of the answer if any.
T3	Holds how many questions are to be worth triple points.
T()	Temporarily holds the triple point value numbers to check against double point value.
TI$	Holds the Commodore internal clock. Used for decrementing the point value on the screen.
TM	Decimal value of the internal timer.
TP	Holds the total starting value point for question on the screen. It is either 25 or 12.
TRUE	Boolean flag indicates that the result is indeed true.
TU	Bolean flag set only if the timer runs out of time. Indicates that the *T*ime is *U*p.
WD$	Used for parsing the answer.
WN	Holds a random number to select from the file.
WRDS	Counter used to count how many words are in your answer.
YF	Boolean flag to indicate whether the user responded with a yes (Y) or a no (N).

Appendix C

The Disk Data Base

PROGRAM LISTING

```
5 REM
6 REM  DISKETTE VERSION - DATA BASE
7 REM
10 POKE 198,0:POKE 649,10:POKE 53280,13:POKE 53281,13
15 BLK$="{39 SP}":BLK$=BLK$+BLK$
20 FALSE=0:TRUE=NOT FALSE:GOTO 810
22 M=54272:FOR L=MTO M+24:POKE L,0:NEXT:POKEM+5,9:
   POKEM+6,0:POKEM+24,15
24 POKEM+1,25:POKEM,30:POKEM+4,33:FORD=1 TO 150:NEXT:
   POKEM+4,32
26 POKE M+24,0:RETURN
30 IF NM<256 THEN HI=0:LO=NM
32 IF NM>=256 THEN HI=INT(NM/256):LO=NM-256*HI
34 RETURN
37 REM
38 REM  RETRIEVE ONE RECORD
39 REM
40 PRINT#1,"P"CHR$(3)CHR$(LO)CHR$(HI)CHR$(1):INPUT#2,Q
   $
41 PRINT#1,"P"CHR$(3)CHR$(LO)CHR$(HI)CHR$(76):INPUT#2,
   A$
42 HD$=Q$+A$:RETURN
43 REM
44 REM  STORE ONE RECORD
45 REM
46 PRINT#1,"P"CHR$(3)CHR$(LO)CHR$(HI)CHR$(1)
47 PRINT#2,LEFT$(TS$,74)
48 PRINT#1,"P"CHR$(3)CHR$(LO)CHR$(HI)CHR$(76)
49 PRINT#2,MID$(TS$,75,37):RETURN
50 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{5 SP}PRESS{2 SP}<RETURN>{2 SP}TO
   {1 SP}CONTINUE"
55 WAIT 197,64,64:GET K$:IF K$<>CHR$(13) THEN 55
58 RETURN
60 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}";LEFT$(BLK$,39)
61 PRINT"{4 SP}F1{1 SP}-{1 SP}SAVE,{1 SP}F3{1 SP}-
   {1 SP}REDO,{1 SP}F5{1 SP}-{1 SP}EXIT"
62 PRINT LEFT$(BLK$,39);:RETURN
70 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";
   LEFT$(BLK$,37):PRINT"{RA}";LEFT$(BLK$,37):RETURN
75 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";LEFT$(BLK$,37
   ):RETURN
77 REM
78 REM  PARSE ANSWER
79 REM
80 WD$="":WRDS=0:FOR I=1 TO LEN(ANS$)
82 IF ASC(MID$(ANS$,I,1))=32 AND WD$="" THEN 90
84 IF ASC(MID$(ANS$,I,1))=32 AND MID$(ANS$,I+1,1)="
   {1 SP}" THEN 90
86 IF ASC(MID$(ANS$,I,1))=32 OR I=LEN(ANS$) THEN WRDS
   =WRDS+1
87 IF WRDS>2 THEN ERR=TRUE:RETURN
88 WD$=WD$+MID$(ANS$,I,1)
90 NEXT I:FOR I=1 TO LEN(WD$):IF ASC(MID$(WD$,I,1))=32
   AND I<>1 THEN 93
```

```
92 NEXT I:RETURN
93 A$=MID$(WD$,1,I):T$=WD$
94 IF A$="THE{1 SP}" OR A$="AN{1 SP}" OR A$="A{1 SP}"
   THEN T$=MID$(WD$,I+1,LEN(T$)-I)
96 ANS$=T$:RETURN
97 REM
98 REM   GET CHARACTERS ROUTINE
99 REM
100 WR=FALSE:RN=1:RW=SP:ANS$="":LOC=0:POKESP+54272,0:
   POKESP,64:POKE SP+1,32:LL=0
102 IF PEEK(198)=0 THEN 102
103 CH=PEEK(631):POKE 198,0
104 IF CH<>13 AND(CH<20 OR (CH>20 AND CH<32) OR CH=44)
    THEN GOSUB 22:GOTO 102
105 IF CH=133 AND SV THEN FLG=2:POKE RW+LOC,32:RETURN
106 IF CH=133 THEN 102
108 IF CH=134 THEN FLG=1:RETURN
110 IF CH=135 AND EX THEN FLG=3:RETURN
112 IF CH=135 THEN 102
114 IF CH=13 AND ANS$<>LEFT$(BLK$,LEN(ANS$)) THEN
   POKE RW+LOC,32:RETURN
116 IF CH=13 THEN POKE RW+LOC,32:GOTO 100
118 IF CH=20 OR CH=157 THEN 140
120 ANS$=ANS$+CHR$(CH):IF CH>31 AND CH<64 THEN 126
122 IF CH>95 AND CH<128 THEN CH=CH-32:GOTO 126
124 CH=CH-64
126 POKE RW+LOC,CH:LOC=LOC+1:POKE RW+LOC+54272,0:POKER
   W+LOC,64:IFLOC<37THEN 102
127 POKE LOC+RW,32:RN=RN+1:IF RN>NR THEN RN=NR:GOTO 16
   4
128 IF PEEK((LOC-1)+RW)=32 THEN WR=FALSE:GOTO 138
130 T$="":FOR I=37 TO 1 STEP -1:IF ASC(MID$(ANS$,I,1))
   =32 THEN 134
132 T$=MID$(ANS$,I,1)+T$:NEXT:RW=RW+80:LOC=0:POKE RW+5
   4272,0:POKE RW,64:GOTO 102
134 ANS$=MID$(MID$(ANS$,1,I)+BLK$,1,37):PRINT"
   {CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";ANS$:
   PRINT"{RA}";T$
135 WR=TRUE:LL=LEN(ANS$)-LEN(T$)
136 ANS$=ANS$+T$:RW=RW+80:LOC=LEN(T$):POKE RW+LOC+5427
   2,0:POKE RW+LOC,64
137 GOTO 102
138 RW=RW+80:POKE RW+54272,0:POKE RW,64:LOC=0:GOTO 102
140 LOC=LOC-1:IF (LOC<=0) AND (RN=1) THEN 100
141 IF LOC>=0 THEN POKE RW+LOC,64
142 POKE RW+LOC+1,32:IF RN=1 THEN 160
144 IF RN=2 AND LOC<0 THEN RN=1:LOC=36:RW=RW-80:POKE R
   W+LOC,64
145 IF RN=2 AND LOC<0 THEN POKE RW+80,1:GOTO 160
146 IF NOT WR THEN 160
148 IF LL+(LEN(ANS$)-37)>=38 THEN 160
150 T$=LEFT$(ANS$,LL):T1$=MID$(ANS$,38,LEN(ANS$)-37)
152 ANS$=T$+T1$:POKE RW+LOC+1,32:LOC=LEN(ANS$)-1:RN=1:
   RW=RW-80
153 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";
   LEFT$(ANS$,LEN(ANS$)-1):PRINT"{RA}";LEFT$(BLK$,37)
154 POKE RW+LOC,64
160 ANS$=LEFT$(ANS$,LEN(ANS$)-1):GOTO 102
164 POKE RW+54309,0:POKE RW+37,64
166 IF PEEK(198)=0 THEN 166
```

```
167 CH=PEEK(631):POKE 198,0
168 IF CH=133 AND SV THEN FLG=2:RETURN
170 IF CH=133 THEN 166
172 IF CH=134 THEN FLG=1:RETURN
174 IF CH=135 AND EX THEN FLG=3:RETURN
176 IF CH=135 THEN 166
178 IF CH=13 AND ANS$<>LEFT$(BLK$,LEN(ANS$)) THEN
   POKE RW+LOC,32:RETURN
180 IF CH=13 THEN 166
182 IF CH=20 OR CH=157 THEN 140
184 GOTO 166
186 WAIT 197,64,64:GET K$:IF K$<>"Y" AND K$<>"N" THEN
   186
187 IF K$="Y" THEN YF=TRUE
188 RETURN
190 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}";LEFT$(BLK$,38)
192 PRINT"{6 SP}PRESS{1 SP}<RETURN>{1 SP}WHEN{1 SP}DON
   E!{7 SP}":PRINTLEFT$(BLK$,38);
193 GOSUB 22:RETURN
194 Q$=MID$(Q$+BLK$,1,74):A$=MID$(A$+BLK$,1,37):QA$=Q$
   +A$:TS$=QA$
195 NM=TN+1:GOSUB 30:GOSUB 46:RETURN
201 REM
202 REM     SCREEN FORM
203 REM
204 PRINT"{SHIFT/CLR-HOME}{DA}{CTRL/BLK}{6 SP}";TT$:
   PRINT"{DA}{CTRL/RVS ON}{40 SP}";
205 FOR I=1 TO 17:PRINT"{CTRL/RVS ON}{1 SP}";SPC(38);"
   {1 SP}";:NEXT I:PRINT "{22 SP}";
210 PRINT"{18 SP}":PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}
   {DA}{RA}QUESTION{1 SP}#":PRINT"{DA}{DA}{RA}";
215 FORI=0TO36:POKE55617+I,0:POKE55697+I,0:POKE 1345+I
   ,120:POKE 1425+I,120:NEXT
220 PRINT"{DA}{DA}{DA}{DA}{DA}ANSWER{1 SP}FOR{1 SP}QUE
   STION{1 SP}#"
222 FOR I=0 TO 36:POKE 55977+I,0:POKE 1705+I,120:NEXT
230 PRINT"{DA}{DA}{DA}{RA}{9 SP}(USE{1 SP}ONLY{1 SP}2
   {1 SP}WORDS){CTRL/BLK}":RETURN
240 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{3 SP}F1{1 SP}-{1 SP}REDO,{3 SP}F3{1 SP}-
   {1 SP}NEXT,{1 SP}F5{1 SP}-{1 SP}PREV"
245 PRINT "{3 SP}F2{1 SP}-{1 SP}DELETE,{1 SP}F4{1 SP}-
   {1 SP}EXIT{13 SP}":RETURN
250 POKE 55508,0:POKE 55879,0:POKE55509,0:POKE 55510,0
   :POKE 55880,0:POKE 55881,0
252 RN$=RIGHT$(BLK$+STR$(NM),3):FOR I=1 TO 3:POKE 1235
   +I,ASC(MID$(RN$,I,1))
254 POKE 1606+I,ASC(MID$(RN$,I,1))
256 NEXT I:RETURN
280 FOR I=1 TO D:NEXT:RETURN
284 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {RA}{11 SP}RE-ENTER{1 SP}ANSWER!":D=600:GOSUB 280
286 GOSUB 22:PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{RA}";LEFT$(BLK$,38):RETURN
287 REM
288 REM    REDO ROUTINE
```

```
289 REM
290 Q$=MID$(QA$,1,74):A$=MID$(QA$,75,37)
300 BOTH=FALSE:PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{2 SP}REDO: {27 SP}"
310 PRINT"{7 SP}Q{1 SP}-{1 SP}QUESTION,{1 SP}A{1 SP}-
    {1 SP}ANSWER{6 SP}":PRINT"{7 SP}B{1 SP}-{1 SP}BOTH,
    {1 SP}";
312 PRINT"X{1 SP}-{1 SP}EXIT{10 SP}";
314 IF PEEK(198)=0 THEN 314
316 CH=PEEK(631):POKE 198,0
320 IF CH<>81 AND CH<>65 AND CH<>66 AND CH<>88 THEN 31
    4
325 IF CH<>88 THEN GOSUB 190:CNG=TRUE
328 IF CH=81 THEN GOSUB 70:GOTO 340
330 IF CH=65 THEN GOSUB 75:GOTO 346
332 IF CH=66 THEN BOTH=TRUE:GOSUB 70:GOSUB 75:GOTO 340
334 IF NOT DE THEN RETURN
336 IF NOT CNG THEN RETURN
338 GOSUB 194:RETURN
340 SV=FALSE:EX=FALSE:FLG=0:NR=2:SP=1305:GOSUB 100
342 IF FLG=1 THEN GOSUB 70:GOTO 340
344 Q$=ANS$:IF NOT BOTH THEN QA$=LEFT$(LEFT$(Q$+BLK$,7
    4)+A$+BLK$,111):GOTO 290
345 Q$=LEFT$(Q$+BLK$,74)
346 SV=FALSE:EX=FALSE:NR=1:FLG=0:SP=1665:GOSUB 100
348 IF FLG=1 THEN GOSUB 75:GOTO 346
350 ERR=FALSE:GOSUB 80:IF ERR THEN GOSUB 284:GOSUB 75:
    GOTO 346
352 A$=ANS$:QA$=LEFT$(Q$+(LEFT$(A$+BLK$,37)),111):
    GOTO 290
397 REM
398 REM    ADDING QUESTIONS
399 REM
400 PRINT"{SHIFT/CLR-HOME}":DE=FALSE
402 TT$="**{2 SP}ADDING{1 SP}QUESTIONS{2 SP}**":GOSUB
    204:GOSUB 60
405 TN=TN+1:IF TN <= 720 THEN 415
408 TN=720:PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{4 SP}MEM
    ORY{1 SP}IS{1 SP}FULL!!!":GOSUB 22:GOSUB 50:RETURN
415 NM=TN:GOSUB 250:GOSUB 70:GOSUB 75
420 NR=2:FLG=0:SV=TRUE:EX=TRUE:SP=1305:GOSUB 100
422 IF FLG=1 THEN GOSUB 70:GOTO 420
424 IF FLG=3 THEN TN=TN-1:RETURN
426 IF FLG=2 AND ANS$=LEFT$(BLK$,LEN(ANS$)) THEN 420
428 Q$=ANS$
430 NR=1:FLG=0:SV=TRUE:EX=TRUE:SP=1665:GOSUB 100
432 IF FLG=1 THEN GOSUB 70:GOSUB 75:GOTO 420
434 IF FLG=3 THEN TN=TN-1:RETURN
435 IF ANS$=LEFT$(BLK$,LEN(ANS$)) THEN 430
436 ERR=FALSE:GOSUB 80:IF ERR THEN  GOSUB 284:GOSUB 75
    :GOTO 430
437 A$=ANS$
438 IF FLG=2 THEN GOSUB 194:GOSUB 22:GOSUB 22:CNG=TRUE
    :POKE 1981,32:GOTO 405
439 GOSUB 22:POKE 56253,0:POKE 1981,64
440 IF PEEK(198)=0 THEN 440
441 CH=PEEK(631):POKE 198,0
442 IF CH<133 OR CH>135 THEN 440
```

```
444 IF CH=133 THEN FLG=2:GOTO 438
446 IF CH=135 THEN TN=TN-1:RETURN
448 QA$=MID$(Q$+BLK$,1,74)+MID$(A$+BLK$,1,37):GOSUB290
   :GOSUB60:GOSUB 22:GOTO 440
497 REM
498 REM   DISPLAY/EDIT QUESTIONS
499 REM
500 IF TN>0 THEN 510
502 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{3 SP}THERE
   {1 SP}ARE{1 SP}NO{1 SP}QUESTIONS{1 SP}AND{1 SP}ANSW
   ERS":PRINT"{DA}{3 SP}IN{1 SP}MEMORY{1 SP}TO{1 SP}";
504 PRINT"DISPLAY/EDIT.":GOSUB 22:GOSUB 50:RETURN
510 ARP=1:DE=TRUE
520 TT$="**{2 SP}DISPLAY/EDIT{1 SP}QUESTIONS{2 SP}**":
   GOSUB 204
522 NM=ARP+1:GOSUB 30:GOSUB 40:QA$=HD$
530 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";
   MID$(QA$,1,37):PRINT"{RA}";MID$(QA$,38,37)
535 PRINT:PRINT"{DA}{DA}{DA}{RA}";MID$(QA$,75,37)
540 NM=ARP:GOSUB 250:GOSUB 240:POKE 56257,0:POKE 1987,
   64
542 IF PEEK(198)=0 THEN 542
543 CH=PEEK(631):POKE 198,0
544 IF CH=133 THEN TT=TN:TN=ARP:GOSUB 290:GOSUB 194:TN
   =TT:GOTO 520
546 IF CH=134 THEN ARP=ARP+1:GOTO 558
548 IF CH=135 THEN ARP=ARP-1:GOTO 566
550 IF CH=137 THEN 576
552 IF CH=138 THEN RETURN
553 GOTO 542
555 REM
556 REM  GET NEXT RECORD
557 REM
558 IF ARP<=TN THEN GOSUB 250:GOTO 522
560 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{3 SP}THERE{1 SP}ARE{1 SP}NO{1 SP}MORE
   {1 SP}QUESTIONS!{8 SP}"
562 PRINTLEFT$(BLK$,38);:GOSUB 22:D=600:GOSUB 280:ARP
   =ARP-1:GOTO 540
563 REM
564 REM  GET PREVIOUS RECORD
565 REM
566 IF ARP>=1 THEN GOSUB 250:GOTO 522
568 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{3 SP}CAN'T{1 SP}GO{1 SP}BELOW{1 SP}RECORD
   {1 SP}#1!{11 SP}"
570 PRINTLEFT$(BLK$,38);:GOSUB 22:D=600:GOSUB 280:ARP
   =ARP+1:GOTO 540
573 REM
574 REM  DELETE RECORD?
575 REM
576 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}";LEFT$(BLK$,39)
578 PRINT"{5 SP}DELETE{1 SP}THIS{1 SP}RECORD?{1 SP}(Y/
   N)":YF=FALSE:GOSUB 22
579 GOSUB 186:IF NOT YF THEN 540
```

```
580 IF TN-1=0 THEN TN=0:GOTO 500
581 IF ARP=TN THEN ARP=ARP-1:GOTO 586
582 NM=TN+1:GOSUB 30:GOSUB 40:TS$=HD$:NM=ARP+1:GOSUB 3
    0:GOSUB 46:
586 TN=TN-1:CNG=TRUE:IF TN<1 THEN 500
588 NM=ARP:GOSUB 250:GOTO 522
597 REM
598 REM    MAIN MENU SELECTION 3 & 4
599 REM
600 NUMQ=TN:CLOSE 2:CLOSE 1:IF NOT CNG AND SL=4 THEN 8
    62
610 IF NOT CNG THEN 810
640 OPEN 1,8,15:OPEN2,8,3,"TRIVIA{1 SP}FILE"
642 PRINT#1,"P"CHR$(3)CHR$(1)CHR$(0)CHR$(1)
650 PRINT#2,NUMQ:CLOSE 2:CLOSE 1
660 IF SL=4 THEN 862
795 REM
800 REM  FIRST MENU
805 REM
810 CNG=FALSE:NUMQ=0:PRINT"{SHIFT/CLR-HOME}{DA}{DA}
    {CTRL/BLK}{8 SP}**{2 SP}TRIVIA{1 SP}DATA{1 SP}BASE
    {2 SP}**"
820 PRINT"{DA}{DA}{DA}{2 SP}DO{1 SP}YOU{1 SP}WANT
    {1 SP}TO:{DA}{DA}":PRINT "{4 SP}1.{1 SP}CREATE
    {1 SP}A{1 SP}NEW{1 SP}TRIVIA{1 SP}FILE"
830 PRINT "{DA}{4 SP}2.{1 SP}LOAD{1 SP}AN{1 SP}EXISTIN
    G{1 SP}FILE":PRINT"{DA}{4 SP}3.{1 SP}EXIT{1 SP}DATA
    {1 SP}BASE{1 SP}PROGRAM"
840 PRINT"{DA}{DA}{4 SP}PLEASE{1 SP}SELECT{1 SP}(1-3)"
    :GOSUB 22
850 WAIT 197,64,64:GET SL$:IF SL$<"1" OR SL$>"3" THEN
    850
860 POKE 1688,ASC(SL$):IF VAL(SL$)<>3 THEN 870
862 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}LOAD"CHR$(34)"ME
    NU"CHR$(34)",8"
864 PRINT "{CLR-HOME}":POKE 198,4:POKE 631,13:POKE 632
    ,82:POKE 633,213:POKE 634,13:END
870 IF VAL(SL$)=1 THEN 900
880 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{3 SP}PLEASE
    {1 SP}INSERT{1 SP}THE{1 SP}DISKETTE"
890 PRINT"{DA}{3 SP}WHERE{1 SP}YOUR{1 SP}QUESTIONS
    {1 SP}AND{1 SP}ANSWERS":PRINT"{DA}{3 SP}HAVE{1 SP}B
    EEN{1 SP}SAVED."
895 GOTO 922
900 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{3 SP}PLEASE
    {1 SP}INSERT{1 SP}A{1 SP}BLANK{1 SP}FORMATTED"
902 PRINT"{DA}{3 SP}DISKETTE{1 SP}WHERE{1 SP}YOU
    {1 SP}WANT{1 SP}YOUR{1 SP}NEW":PRINT"{DA}{3 SP}TRIV
    IA{1 SP}FILE{1 SP}SAVED."
922 GOSUB 22:GOSUB 50
924 OPEN 1,8,15:OPEN 2,8,3,"TRIVIA{1 SP}FILE,L,"+CHR$(
    114)
925 IF VAL(SL$)=1 THENPRINT#1,"P"CHR$(3)CHR$(1)CHR$(0)
    CHR$(1):PRINT#2,0
932 PRINT#1,"P"CHR$(3)CHR$(1)CHR$(0)CHR$(1)
940 INPUT#2,NUMQ
945 CLOSE 2:CLOSE 1
997 REM
```

```
998 REM     DB MAIN MENU
999 REM
1000 TN=NUMQ:OPEN 1,8,15:OPEN 2,8,3,"TRIVIA{1 SP}FILE"
1005 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{CTRL/BLK}{7 SP}**
   {2 SP}TRIVIA{1 SP}DB{1 SP}MAIN{1 SP}MENU{2 SP}**"
1010 PRINT"{DA}{DA}{DA}{7 SP}1.{1 SP}ADD{1 SP}QUESTION
   S":PRINT"{DA}{7 SP}2.{1 SP}DISPLAY/EDIT{1 SP}QUESTI
   ONS"
1020 PRINT"{DA}{7 SP}3.{1 SP}CHANGE{1 SP}FILES":PRINT"
   {DA}{7 SP}4.{1 SP}EXIT{1 SP}PROGRAM"
1030 PRINT "{DA}{DA}{7 SP}PLEASE{1 SP}SELECT{1 SP}(1-4
   )":GOSUB 22
1040 WAIT 197,64,64:GET SL$:IF SL$<"1" OR SL$>"4"
   THEN 1040
1045 POKE 55923,0:POKE 1651,ASC(SL$):SL=VAL(SL$):ON SL
    GOSUB 400,500,600,600
1050 GOTO 1005
```

VARIABLE LISTING

The following is a listing of the main variables in the disk data base. It includes the most important variables and may not include all the variables in the program.

A$	Temporarily holds the answer before packing and concatenating to the question.
ANS$	Holds the answer. Maximum length is 37 characters.
ARP	Holds the current location of the array QA$ for displaying the record to the screen. Used in display/edit menu.
BLK$	Blank string the size of the maximum size field (74). Used for packing the question and answer to their maximum length.
BOTH	Boolean flag set to indicate if the user wants to REDO both the question and the answer. If true, the user wants to edit both parts of the record.
CH	Holds the ASCII value of the character that was pressed. The ASCII value is taken from the buffer location at 631.

CNG	Boolean flag. Indicates if any change has been made to the existing file in memory.
DE	Flag set to indicate whether the program execution came from the *D*isplay/*E*dit menu. If true, it did. Makes sure you return to display/edit menu.
ERR	Flag used to indicate if an error occurred within a procedure. This type of an error is not a system error but a user error (invalid response, or illegal characters, etc.).
EX	Boolean flag. Indicates whether or not to allow the user to use the **F5** or **F4** key.
FALSE	Boolean flag set to Ø. Indicates a false value.
FLG	Flag set to indicate if one of the command keys (**F1**, **F3**, etc.) was pressed. Returns a 1 if **F3** was pressed, a 2 if **F1** was pressed, and a 3 if **F5** was pressed.
HD$	Holds the string concatenation of the question and the answer after they have been loaded into memory.
HI	Holds the high byte calculated from the record number to load or save to disk.
I,D,L	Looping variables for FOR . . . NEXT statements.
K$	Used in getting a key from the keyboard via GET statement.
LL	Length of the first line on the question. Used to determine when the question needs to backwrap around to the beginning line.
LOC	Location of the cursor on the current row.
LO	Holds the low byte calculated by the record number to retrieve or save to disk.
M	Sound chip beginning address location.

NM	Record number to print. Same as the question number. Shows what question the user is currently entering.
NR	The number of rows the cursor is allowed to make. For the question, NR is set to 2; for the answer, it is set to 1.
NUMQ	Number of questions currently loaded into memory or initially set to Ø for creating a new file.
QA$	Holds the questions and answers in a concatenated string. Total string length is 111 characters—74 for the question and 37 for the answer.
Q$	Holds the question. Maximum length is 74 characters.
RN	Counter to keep track of how many rows the cursor has moved compared to how many it is allowed to move.
RW	Is used for keeping the cursor on the same line until the end of the line is reached, then move it down to the next line.
SL	Holds the selection number the user chose from the main menu.
SL$	Holds the string representation of the number pressed from one of the menus.
SP	Starting row location to place the cursor and where to start printing characters as they are typed.
SV	Boolean flag. Indicates whether or not to allow the user to press **F1** while typing in the question or answer.
T$	Temporarily holds the answer after it is converted to all uppercase letters.

T1$	Temporarily holds the wraparound answer to the question.
TN	Holds the array number of the next location to store the question and answer.
TRUE	Boolean flag indicates that the result is indeed true.
TS$	Holds the concatenated string to store to disk.
TT	Temporarily holds the total number of questions and answers in memory.
TT$	Holds the title string to be printed at the top of each screen.
WD$	Temporarily holds the new answer after parsing.
WR	Holds a true if the line wraps around to the next line, or false if the question didn't wrap around. Used in getting the question.
WRDS	Counter used to count how many words are in your answer.
YF	Boolean flag to indicate whether the user responded with a yes (Y) or a no (N).

Appendix D

The Disk Game

PROGRAM LISTING

```
5 REM
6 REM  DISKETTE VERSION - GAME
7 REM
10 DIM QA$(30),QNU(30),CK(30),ND(8),NT(4):POKE 53280,1
   3:POKE 53281,13
11 BLK$="{37 SP}":FALSE=0:TRUE=NOT FALSE
12 M=54272:FOR L=MTO M+24:POKE L,0:NEXT:GOTO 800
15 RN$=RIGHT$(BLK$+STR$(TM),2):POKE 55487,0:POKE 55488
   ,0
16 POKE 1215,ASC(MID$(RN$,1,1)):POKE 1216,ASC(MID$(RN$
   ,2,1)):RETURN
22 M=54272:POKE M+24,15:POKEM+5,9:POKEM+6,0:POKE M+1,2
   5:POKE M,30
24 POKE M+4,33:FOR I=1 TO 150:NEXT:POKE M+4,32:POKE M
   +24,0:RETURN
30 M=54272:POKE M+24,15:POKEM+5,9:POKEM+6,0:POKE M+1,6
   :POKE M,71
32 POKE M+4,33:FOR I=1 TO 150:NEXT:POKE M+4,32:POKE M
   +24,0:RETURN
33 REM
34 REM  CALCULATES HI&LO BYTES OF REC. #
35 REM
36 IF NM<256 THEN HI=0:LO=NM
37 IF NM>=256 THEN HI=INT(NM/256):LO=NM-256*HI
38 RETURN
39 REM
40 REM  RETRIEVE RECORD
41 REM
42 PRINT#1,"P"CHR$(3)CHR$(LO)CHR$(HI)CHR$(1):INPUT#2,Q
   $
44 PRINT#1,"P"CHR$(3)CHR$(LO)CHR$(HI)CHR$(76):INPUT#2,
   A$
46 HD$=Q$+A$:RETURN
50 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{5 SP}PRESS{2 SP}<RETURN>{2 SP}TO
   {1 SP}CONTINUE";
55 WAIT 197,64,64:GET RT$:IF RT$<>CHR$(13) THEN 55
58 RETURN
60 FOR I=1 TO D:NEXT:RETURN
61 REM
62 REM   DISPLAY STATUS FOR A PLAYER
63 REM
64 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{RA}{RA}
   {RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}";PL;"{RA}{RA}{RA}{RA}
   {RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}";LEFT$(PN$(PL)
   ,10)
65 FOR Q=0 TO 3:POKE 1832+Q,32:POKE 1849+Q,32:NEXT:
   POKE 1853,32:POKE 1854,32
66 POKE 1855,32:POKE 1856,32:POKE 1857,32:POKE 1858,32
68 PRINT"{DA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}";SC(PL);"
   {RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}{RA}
   {RA}{RA}{RA}";NC(PL);"/";NQ(PL):RETURN
72 REM
73 REM   ERASES ANSWER
74 REM
75 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{RA}";BLK$:RETURN
```

```
77 REM
78 REM   PARSE ANSWER
79 REM
80 WD$="":WRDS=0:FOR I=1 TO LEN(ANS$)
82 IF ASC(MID$(ANS$,I,1))=32 AND WD$="" THEN 90
84 IF ASC(MID$(ANS$,I,1))=32 AND MID$(ANS$,I+1,1)="
   {1 SP}" THEN 90
86 IF ASC(MID$(ANS$,I,1))=32 OR I=LEN(ANS$) THEN WRDS
   =WRDS+1
87 IF WRDS>2 THEN ERR=TRUE:RETURN
88 WD$=WD$+MID$(ANS$,I,1)
90 NEXT I:FOR I=1 TO LEN(WD$):IF ASC(MID$(WD$,I,1))=32
    AND I<>1 THEN 93
92 NEXT I:RETURN
93 A$=MID$(WD$,1,I):T$=WD$
94 IF A$="THE{1 SP}" OR A$="AN{1 SP}" OR A$="A{1 SP}"
   THEN T$=MID$(WD$,I+1,LEN(T$)-I)
96 ANS$=T$:RETURN
97 REM
98 REM   GET ANSWER ROUTINE
99 REM
100 POKE 198,0:POKE 649,1:TI$="000000"
101 RW=SP:ANS$="":LOC=0:POKESP+54272,0:POKESP,64:POKES
   P+1,32
102 IFPEEK(198)=0THENTM=VAL(RIGHT$(TI$,2)):TM=TP-TM:
   IFTM<0 THEN TU=TRUE:RETURN
103 GOSUB 15:IF PEEK(198)=0 THEN 102
104 CH=PEEK(631):POKE 198,0
106 IF CH<>13 AND (CH<20 OR (CH>20 AND CH<32)) THEN 10
   2
109 IF CH=133 THEN EX=TRUE:RETURN
112 IF CH=13 AND ANS$<>"" THEN POKE RW+LOC,32:RETURN
113 IF CH=13 THEN 102
114 IF CH=20 OR CH=157 THEN 138
115 TM=VAL(RIGHT$(TI$,2)):TM=TP-TM:IF TM<0 THEN TU=TRU
   E:RETURN
116 GOSUB 15
117 ANS$=ANS$+CHR$(CH):IF CH>31 AND CH<64 THEN 122
118 IF CH>95 AND CH<128 THEN CH=CH-32:GOTO 122
120 CH=CH-64
122 POKE RW+LOC,CH:LOC=LOC+1:POKE RW+LOC+54272,0:POKER
   W+LOC,64:IFLOC<37THEN 102
124 POKE LOC+RW,32:POKE RW+54309,0:POKE RW+37,64
130 IF PEEK(198)=0 THENTM=VAL(RIGHT$(TI$,2)):TM=TP-TM:
   IFTM<0THENTU=TRUE:RETURN
131 GOSUB 15:IF PEEK(198)=0 THEN 130
132 IF CH=13 THEN RETURN
134 IF CH=20 OR CH=157 THEN 138
135 IF CH=133 THEN EX=TRUE:RETURN
136 GOTO 130
138 LOC=LOC-1:IF LOC<1 THEN 101
140 POKE RW+LOC,64:POKE RW+LOC+1,32
141 TM=VAL(RIGHT$(TI$,2)):TM=30-TM:IF TM<=0 THEN TU=TR
   UE:RETURN
142 GOSUB 15
144 ANS$=LEFT$(ANS$,LEN(ANS$)-1):GOTO 102
183 REM
184 REM   Y OR N ROUTINE
185 REM
186 WAIT 197,64,64:GET K$:IF K$<>"Y" AND K$<>"N" THEN
   186
```

```
187 IF K$="Y" THEN YF=TRUE
188 RETURN
197 REM
198 REM   GAME PLAYING
199 REM
200 GOSUB 64:NM=AL:GOSUB 450:TM=25:GOSUB 15:TP=25
202 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{RA}";
   LEFT$(QA$(AL),37):PRINT"{RA}";MID$(QA$(AL),38,37)
203 PS=PL
204 PD=FALSE:FOR I=1 TO DB:IF ND(I)=QNU(AL) THEN PD=TR
   UE:GOTO 210
206 NEXT:PD=FALSE:FOR I=1 TO T3:IF NT(I)=QNU(AL) THEN
   PT=TRUE:GOTO 210
208 NEXT I:PT=FALSE
210 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{RA}";LEFT$(BLK$,37):GOSUB 75
212 IF PD THEN PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{RA}{10 SP}DOUBLE{1 SP}VALUE!":
   GOSUB 22:GOSUB 22
214 IF PT THEN PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{DA}{DA}{RA}{10 SP}TRIPLE{1 SP}VALUE!":
   GOSUB 22:GOSUB 22
220 TU=FALSE:EX=FALSE:SP=1505:GOSUB 100:PI=TM
222 IF EX THEN 600
224 IF TU THEN 240
226 IF PD THEN PI=PI*2
228 IF PT THEN PI=PI*3
230 GOSUB 80:ANS$=MID$(ANS$+BLK$,1,37):CA$=MID$(QA$(AL
   ),75,37)
232 IF CA$=ANS$ THEN 250
240 GOSUB 30:GOSUB 30:D=300:GOSUB 60:NQ(PL)=NQ(PL)+1:P
   L=PL+1:IF PL>NP THEN PL=1
242 IF PL<>PS THEN TP=12:GOSUB 15:GOSUB 64:GOSUB 75:
   GOTO 220
244 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{RA}THE{1 SP}CORRECT{1 SP}ANSWER{1 SP}IS:":
   GOSUB 30
245 PRINT"{DA}{RA}";MID$(QA$(AL),75,37):D=1000:GOSUB 6
   0
246 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
   {DA}{DA}{RA}ANSWER{19 SP}"
248 GOSUB 75:GOTO 260
250 GOSUB 22:GOSUB 22:SC(PL)=SC(PL)+PI:NC(PL)=NC(PL)+1
   :NQ(PL)=NQ(PL)+1
252 GOSUB 64:D=500:GOSUB 75:GOSUB 60
260 AL=AL+1:IF AL>MN THEN 280
262 PL=PS+1:IF PL>NP THEN PL=1
264 PD=FALSE:PT=FALSE:GOTO 200
280 D=400:GOSUB 64:GOSUB 60:GOTO 750
450 POKE 55468,0:POKE 55469,0
452 RN$=RIGHT$(BLK$+STR$(NM),2):POKE 1196,ASC(MID$(RN$
   ,1,1))
456 POKE 1197,ASC(MID$(RN$,2,1)):RETURN
497 REM
498 REM   CHECKS FOR DUPLICATE NUMBERS
499 REM
500 IF X=0 THEN ERR=FALSE:RETURN
502 FOR Q=1 TO X:IF NU=CK(Q) THEN ERR=TRUE:RETURN
504 NEXT:ERR=FALSE:RETURN
507 REM
508 REM   SELECT & LOAD RANDOM QUESTIONS
509 REM
```

```
510 OPEN 1,8,15:OPEN 2,8,3,"TRIVIA{1 SP}FILE"
512 PRINT#1,"P"CHR$(3)CHR$(1)CHR$(0)CHR$(1)
514 INPUT#2,NUMQ:IF NUMQ>30 THEN MN=30:GOTO 522
516 MN=NUMQ:FOR I=1 TO MN
518 WN=INT(RND(0)*(MN-1))+2:NU=WN:ERR=FALSE:X=I-1:
    GOSUB 500:IF ERR THEN 518
520 CK(I)=WN:QNU(I)=WN:NEXT:GOTO 530
522 FOR I=1 TO 30
524 WN=INT(RND(0)*(NUMQ-1))+2:NU=WN:ERR=FALSE:X=I-1:
    GOSUB 500:IF ERR THEN 518
526 CK(I)=WN:QNU(I)=WN:NEXT
530 FOR I=1 TO MN:NM=QNU(I):GOSUB 36:GOSUB 42:QA$(I)=Q
    $+A$:NEXT
532 CLOSE 2:CLOSE 1:RETURN
597 REM
598 REM   QUIT GAME EARLY?
599 REM
600 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{RA}{9 SP}ARE{1 SP}YOU{1 SP}SURE?{1 SP}(Y/N
    )":GOSUB 22
610 YF=FALSE:GOSUB 186:IF NOT YF THEN PRINT"
    {CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {RA}";
620 IF NOT YF THEN PRINT"{13 SP}F1{1 SP}-{1 SP}QUIT
    {7 SP}":TP=PI:GOSUB 75:GOTO 220
747 REM
748 REM  END OF GAME STATUS
749 REM
750 GOSUB 22:PRINT"{SHIFT/CLR-HOME}{DA}{DA}{9 SP}**
    {1 SP}FINAL{1 SP}STATS{1 SP}**":PRINT"{DA}{DA}{DA}
    {3 SP}NAME{8 SP}";
752 PRINT"{1 SP}SCORE{9 SP}NC/NQ":PRINT"{CTRL/RVS ON}
    {39 SP}"
758 FOR I=1 TO NP:PN(I)=I:NEXT:FOR J=1 TO NP-1:J1=J
760 FOR K=J+1 TO NP:IF SC(J1)<SC(K) THEN J1=K
762 NEXT K:IF J1<>J THEN T=SC(J):SC(J)=SC(J1):SC(J1)=T
    :T=PN(J):PN(J)=PN(J1)
764 IF J1<>J THEN PN(J1)=T
766 NEXT J
768 FOR I=1 TO NP:PRINTLEFT$(PN$(PN(I)),10);"{7 SP}";S
    C(I);"{6 SP}";
769 PRINT NC(PN(I));"/";NQ(PN(I)):NEXT
770 IF EX THEN D=2000:GOSUB 60:GOTO 776
772 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}ANOTHER{1 SP}GAME?{1 SP}(Y/N)":GOSUB 22:YF=FALS
    E
774 GOSUB 186:IF YF THEN X=RND(-1):GOTO 860
776 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}LOAD"CHR$(34)"ME
    NU"CHR$(34)",8"
778 PRINT"{CLR-HOME}":POKE 198,4:POKE 631,13:POKE 632,
    82:POKE 633,213:POKE 634,13:END
797 REM
798 REM   INTRODUCTION SCREEN
799 REM
800 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}
    {CTRL/BLK}{3 SP}WELCOME{1 SP}TO{1 SP}COMMODORE
    {1 SP}64{1 SP}TRIVIA{1 SP}GAME"
810 PRINT"{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}{DA}{DA}{DA}{DA}{6 SP}NEED{1 SP}INSTRUCTION
    S?{1 SP}(Y/N)":YF=FALSE:GOSUB 22
```

```
820 GOSUB 22:GOSUB 186:IF NOT YF THEN 860
822 PRINT"{SHIFT/CLR-HOME}{CTRL/BLK}{9 SP}INSTRUCTIONS
   "
824 PRINT"{DA}{2 SP}THE{1 SP}OBJECT{1 SP}OF{1 SP}THE
   {1 SP}GAME{1 SP}IS{1 SP}TO{1 SP}GAIN{1 SP}AS":
   PRINT"MANY{1 SP}POINTS{1 SP}AS";
826 PRINT"{1 SP}POSSIBLE{1 SP}BY{1 SP}ANSWERING":
   PRINT"RANDOM{1 SP}TRIVIA{1 SP}QUESTIONS."
828 PRINT"{DA}{2 SP}EACH{1 SP}QUESTION{1 SP}IS{1 SP}WO
   RTH{1 SP}A{1 SP}STARTING":PRINT"VALUE{1 SP}OF
   {1 SP}25{1 SP}POINTS.";
830 PRINT"{2 SP}THE{1 SP}FASTER{1 SP}YOU":PRINT"ANSWER
   {1 SP}THE{1 SP}QUESTION{1 SP}THE{1 SP}MORE{1 SP}POI
   NTS";
832 PRINT"{1 SP}YOU{1 SP}CAN{1 SP}OBTAIN.{2 SP}AS
   {1 SP}TIME{1 SP}GOES{1 SP}BY,{1 SP}THE"
834 PRINT "POINT{1 SP}VALUE{1 SP}DECREASES.":PRINT"
   {DA}{2 SP}IF{1 SP}A{1 SP}PLAYER{1 SP}MISSES{1 SP}TH
   E{1 SP}ANSWER,"
836 PRINT"THE{1 SP}NEXT{1 SP}PLAYER{1 SP}IN{1 SP}LINE
   {1 SP}GETS{1 SP}A{1 SP}CHANCE":PRINT"TO{1 SP}ANSWER
   {1 SP}THE{1 SP}QUESTION.";
838 PRINT"{2 SP}ONCE{1 SP}ALL{1 SP}THE":PRINT"PLAYERS
   {1 SP}HAVE{1 SP}HAD{1 SP}A{1 SP}CHANCE{1 SP}TO
   {1 SP}ANSWER"
840 PRINT"THE{1 SP}QUESTION,{1 SP}THEN{1 SP}THE{1 SP}C
   ORRECT{1 SP}ANSWER":PRINT"IS{1 SP}DISPLAYED."
842 PRINT"{DA}{2 SP}THE{1 SP}GAME{1 SP}CAN{1 SP}BE
   {1 SP}PLAYED{1 SP}BY{1 SP}1-4":PRINT"PLAYERS.";
844 PRINT"{3 SP}GOOD{1 SP}LUCK!...HAVE{1 SP}FUN!"
846 PRINT"{DA}{CTRL/BLK}*{1 SP}ALL{1 SP}ANSWERS{1 SP}M
   UST{1 SP}MATCH{1 SP}EXACTLY!"
850 GOSUB 50
860 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{2 SP}PL
   EASE{1 SP}INSERT{1 SP}THE{1 SP}DISKETTE{1 SP}THAT
   {1 SP}THE":PRINT"{2 SP}QUESTIONS{1 SP}AND";
862 PRINT"{1 SP}ANSWERS{1 SP}ARE{1 SP}STORED{1 SP}ON."
864 GOSUB 22:GOSUB 50
866 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}
   {1 SP}PLEASE{1 SP}LEAVE{1 SP}YOUR{1 SP}TRIVIA
   {1 SP}DISKETTE"
868 PRINT"{DA}{1 SP}IN{1 SP}THE{1 SP}DISK{1 SP}DRIVE
   {1 SP}WHILE{1 SP}I{1 SP}LOAD{1 SP}THE"
870 PRINT"{DA}{1 SP}QUESTIONS{1 SP}AND{1 SP}ANSWERS.":
   GOSUB 50:PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{2 SP}LO
   ADING{1 SP}QUESTIONS..."
872 GOSUB 510
882 REM
883 REM    SELECT QUESTIONS (DBL & TRP POINTS)
884 REM
885 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{5 SP}PLEASE
   {1 SP}WAIT...I'M{1 SP}WORKING"
920 DB=INT(MN*.25)+1:T3=INT(MN*.10)+1:FOR I=1 TO DB
924 WN=INT(RND(0)*MN)+1:NU=QNU(WN):X=I-1:ERR=FALSE:
   GOSUB 500:IF ERR THEN 924
926 ND(I)=QNU(WN):CK(I)=QNU(WN):NEXT I:FOR I=1 TO T3:C
   K(I)=0:NEXT:FOR I=1 TO T3
928 WN=INT(RND(0)*MN)+1:NU=QNU(WN):X=I-1:ERR=FALSE:
   GOSUB 500:IF ERR THEN 928
929 FOR J=1 TO T3:T(J)=CK(J):NEXT
930 FOR J=1 TO DB:CK(J)=ND(J):NEXT:ERR=FALSE:X=ND:
   GOSUB 500
932 IF ERR THEN 928
```

```
933 FOR J=1 TO T3:CK(J)=T(J):NEXT
934 NT(I)=QNU(WN):CK(I)=QNU(WN):NEXT I
935 REM
936 REM   NUMBER OF PLAYERS AND NAMES
937 REM
940 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{5 SP}NU
    MBER{1 SP}OF{1 SP}PLAYERS{1 SP}(1-4)":GOSUB 22
942 WAIT 197,64,64:GET NP$:IF NP$="" THEN 942
944 IF ASC(NP$)<49 OR ASC(NP$)>52 THEN 942
946 POKE55526,0:POKE 1254,ASC(NP$):NP=ASC(NP$)-48:FOR
    I=1 TO NP
947 PRINT"{SHIFT/CLR-HOME}{DA}{DA}{DA}{DA}{DA}{DA}
    {DA}{DA}"
948 PRINT"PLAYER{1 SP}#{1 SP}";I:PN$(I)="":INPUT "
    {DA}{DA}{2 SP}NAME";PN$(I)
950 IF PN$(I)="" THEN 947
952 PN$(I)=MID$(PN$(I)+BLK$,1,10):NEXT I
957 REM
958 REM   DISPLAY GAME SCREEN
959 REM
960 PRINT"{SHIFT/CLR-HOME}{DA}{CTRL/BLK}{10 SP}**
    {1 SP}TRIVIA{1 SP}GAME{1 SP}**"
961 PRINT"{DA}{CTRL/RVS ON}{40 SP}";
962 FOR I=1 TO 13:PRINT"{CTRL/RVS ON}{1 SP}";SPC(38);"
    {1 SP}";:NEXT
964 PRINT"{40 SP}";
966 FOR I=1 TO 5:PRINT"{1 SP}";SPC(38);"{1 SP}";:NEXT:
    PRINT"{20 SP}";
967 PRINT"{20 SP}{CTRL/RVS OFF}";
970 PRINT"{CLR-HOME}{DA}{DA}{DA}{DA}{RA}QUESTION
    {1 SP}#":PRINT"{DA}{DA}{DA}{DA}{DA}{RA}ANSWER:"
972 FOR I=0 TO 36:POKE 55817+I,0:POKE 1545+I,120:NEXT
974 PRINT"{DA}{DA}{DA}{RA}{9 SP}(USE{1 SP}ONLY{1 SP}2
    {1 SP}WORDS){CTRL/BLK}"
976 PRINT"{DA}{DA}{DA}{RA}PLAYER{1 SP}#{10 SP}NAME:":
    PRINT"{DA}{RA}SCORE: {11 SP}NC/NQ:"
978 PRINT"{DA}{RA}{CTRL/BLK}{13 SP}F1{1 SP}-{1 SP}QUIT
    {1 SP}":AL=1:PL=1:FOR I=1 TO NP:SC(I)=0
980 NC(I)=0:NQ(I)=0:NEXT I:GOTO 200
```

VARIABLE LISTING

The following is a listing of the main variables in the disk game. It includes the most important variables and may not include all the variables in the program.

A$	Holds the first word of the answer to see if it matches "a", "an", or "the". Also holds the answer loaded into memory from disk.
AL	Holds the array number of the question that is displayed.
ANS$	Holds the answer that the player has typed. Maximum length is 37 characters.
BLK$	Blank string the size of the maximum size field (37). Used for packing the answer to its maximum length.
CA$	Holds the correct answer. Used for comparing the two answers for equality.
CH	Holds the ASCII value for one character at a time.
CK()	Holds the number for checking against duplicate numbers when selecting random questions, double and triple point value.
DB	Holds the number of double point questions that this game has.
ERR	Flag used to indicate if an error occurred within a procedure.
EX	Holds the boolean flag for quitting the game early.
FALSE	Boolean flag set to Ø. Indicates a false value.
I,J,K,L,D & Q	Looping variables for FOR . . . NEXT statements.
HI	Holds the high byte of the record number to retrieve from the disk.

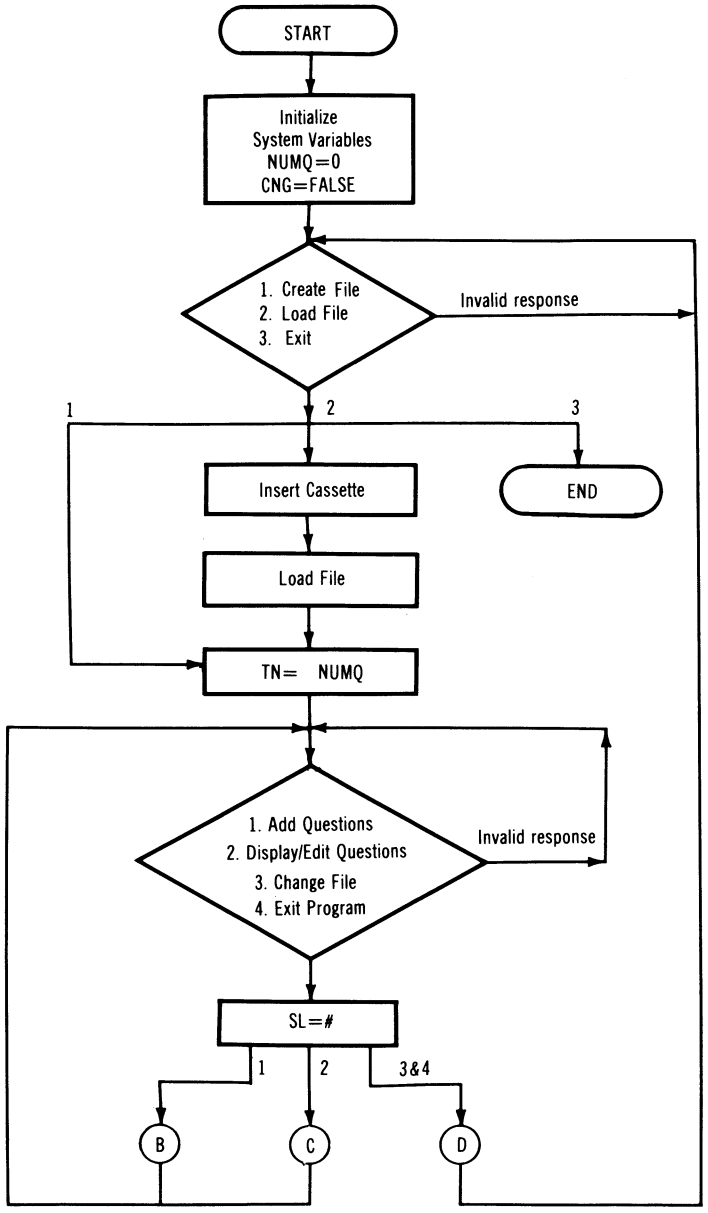
K$	Holds the key pressed using GET statement.
LO	Holds the low byte of the record number to retrieve from the disk.
LOC	Location on the answer line of the cursor.
M	Holds the sound chip memory location.
MN	Maximum number of questions to use. 30 if NUMQ>30 or is the same as NUMQ.
NC()	Holds the number of correct questions that the player answered.
ND(8)	Holds the randomly generated numbers that are to be assigned double point values.
NM	Holds the question number to be printed to the screen.
NP	The total number of players playing the game (from one to four).
NQ()	Holds the number of questions that each player was asked.
NT()	Holds the numbers for the triple point questions.
NUMQ	Number of questions currently loaded into memory from the cassette file.
PD	Boolean flag to indicate if the current question is to be double value or not. Used for printing the DOUBLE VALUE message.
PI	Holds the number of points the player gets when answering the question correctly.
PL	Holds the current player number to display on the screen.
PN$()	Holds the players names. Up to 10 characters long.

PS	Holds the player number of who started first. This keeps the player going in the same order no matter who answers a missed question.
PT	Holds the triple value flag set for printing the TRIPLE VALUE message or not.
Q$	Holds the question loaded into memory from the tape.
QA$(3Ø)	Holds the questions and answers in a concatenated string. Total string length is 111 characters. 74 for the question and 37 for the answer.
QNU(3Ø)	Holds the randomly generated sequence of numbers in which to display the questions in a random format.
RN$	Holds the string value of the internal timer before printing the clock.
RW	Starting row location to place the cursor and where to start printing characters as they are typed.
SC()	Holds the players' score.
SP	Starting cursor position on the screen for entering the answer.
T$	Holds the word to parse out of the answer, if any.
T()	Temporarily holds the triple point value numbers to check against double point value.
T3	Holds how many questions are to be triple pointers.
TI$	Holds the Commodore internal clock. Used for decrementing the point value on the screen.
TM	Decimal value of the internal timer.
TP	Holds the total starting value point for the question on the screen. It is either 25 or 12.
TRUE	Boolean flag indicates that the result is indeed true.

TU	Boolean flag set only if the timer runs out of time. Indicates that the *T*ime is *U*p.
WD$	Used for parsing the answer.
WN	Holds a random number to select from the file.
WRDS	Counter used to count how many words are in your answer.
YF	Boolean flag to indicate whether the user responded with a yes (Y) or a no (N).

Flowchart 1 — Cassette Data Base (foldout)

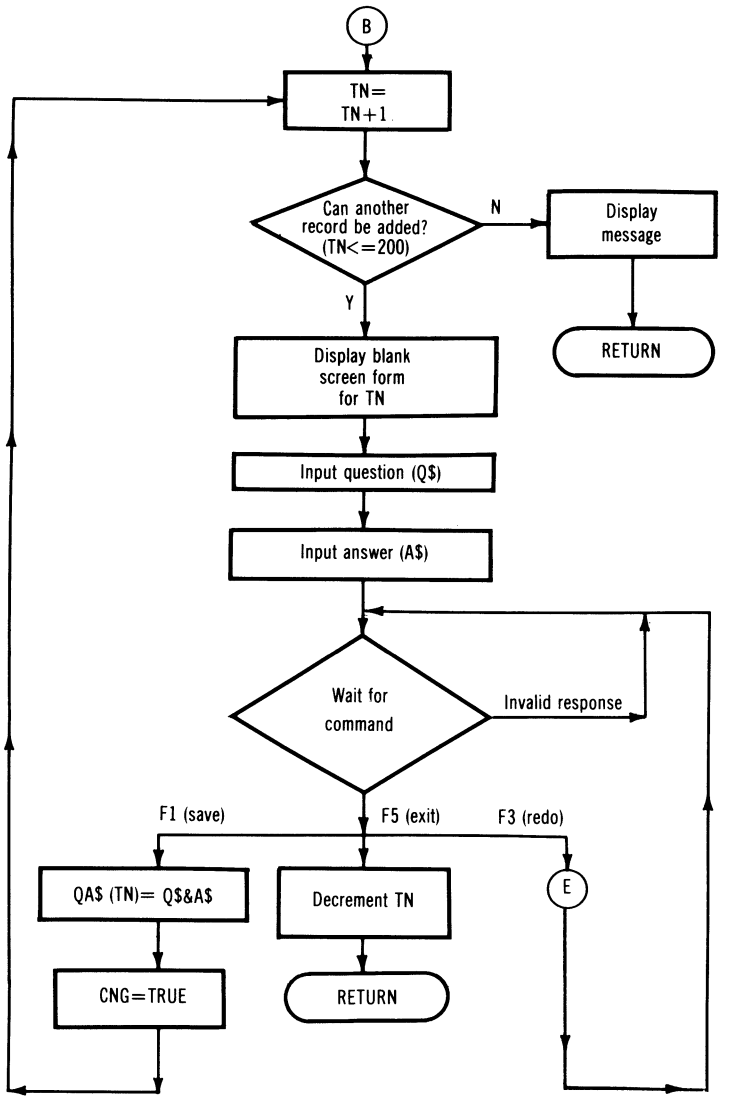
Flowchart 1 — Cassette Data Base



Flowchart 1 cont. — Cassette Data Base

(foldout)

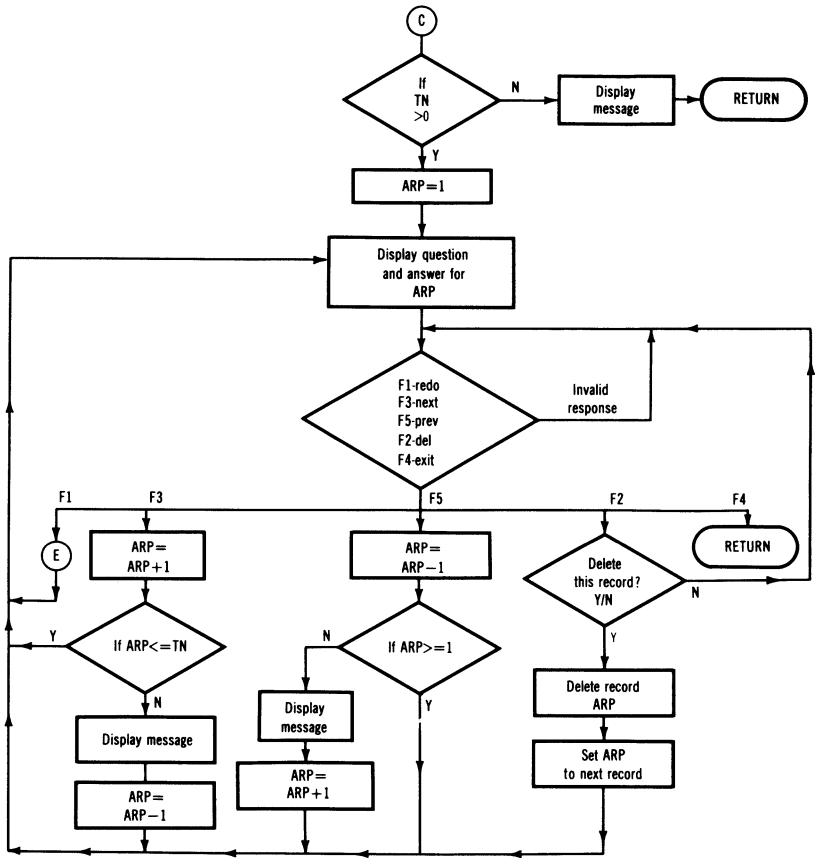
Flowchart 1 cont. — Cassette Data Base



Flowchart 1 cont. — Cassette Data Base

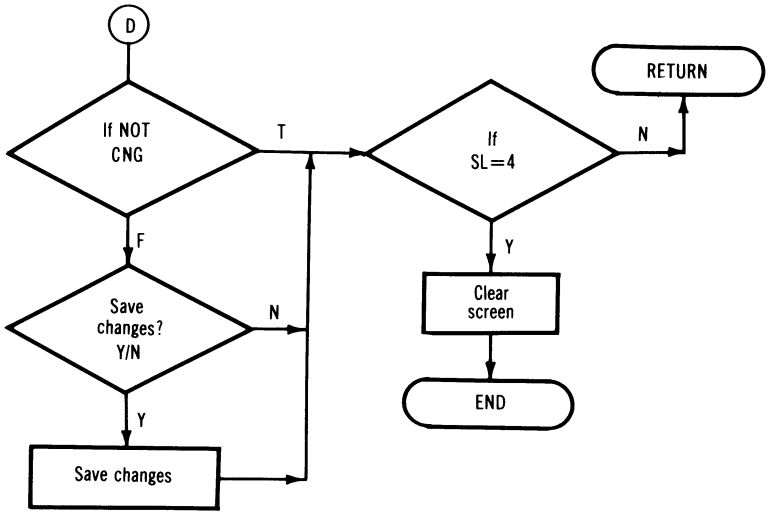
(foldout)

Flowchart 1 cont. — Cassette Data Base



Flowchart 1 cont. — Cassette Data Base
(foldout)

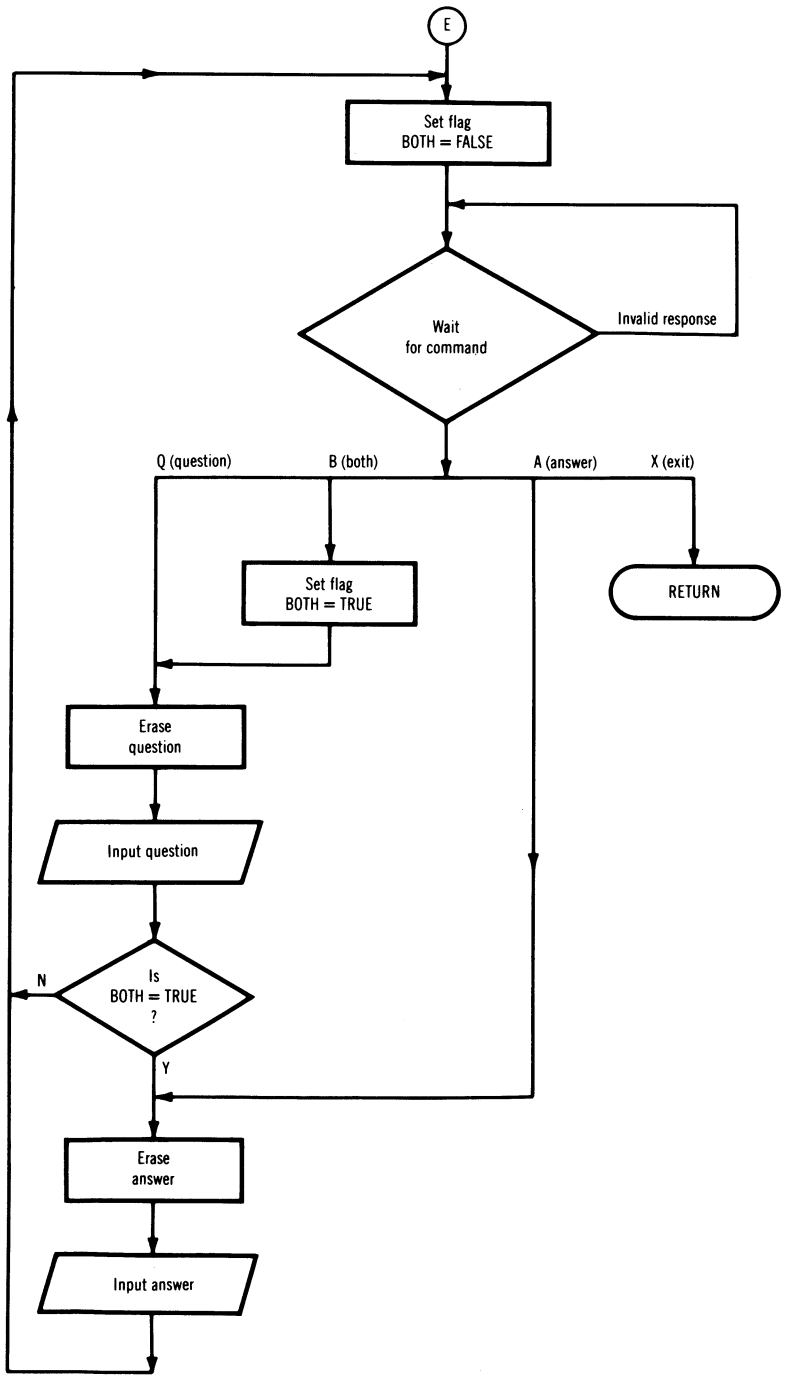
Flowchart 1 cont. — Cassette Data Base



Flowchart 2 cont. — Cassette Data Base

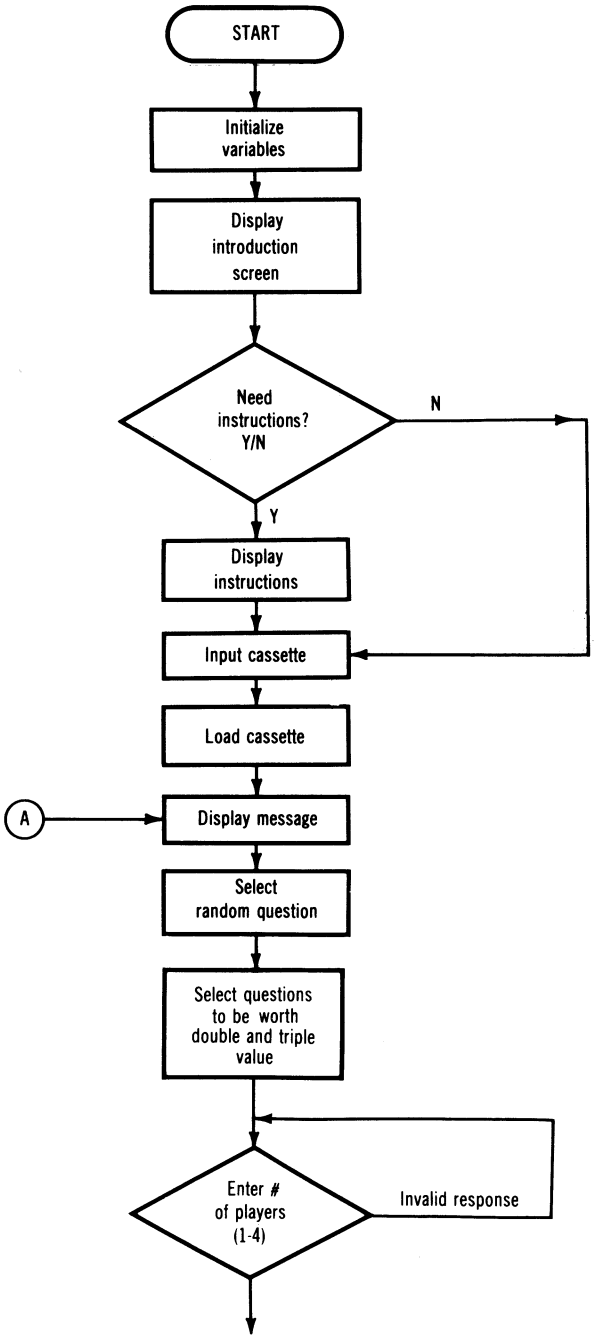
(foldout)

Flowchart 1 cont. — Cassette Data Base



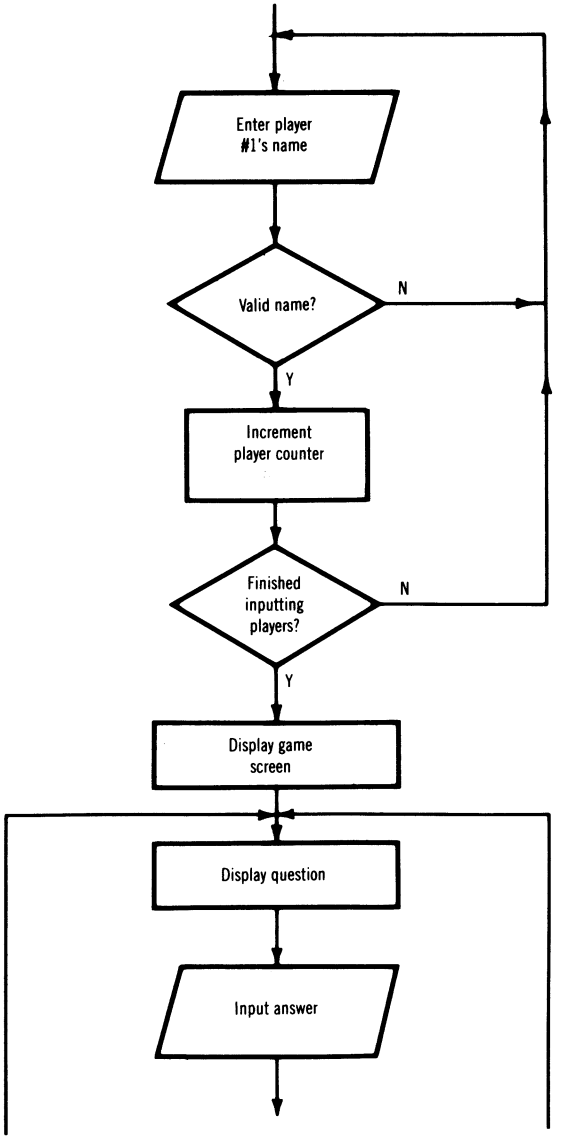
Flowchart 2 — Cassette Game
(foldout)

Flowchart 2 — Cassette Game



Flowchart 2 cont. — Cassette Game
(foldout)

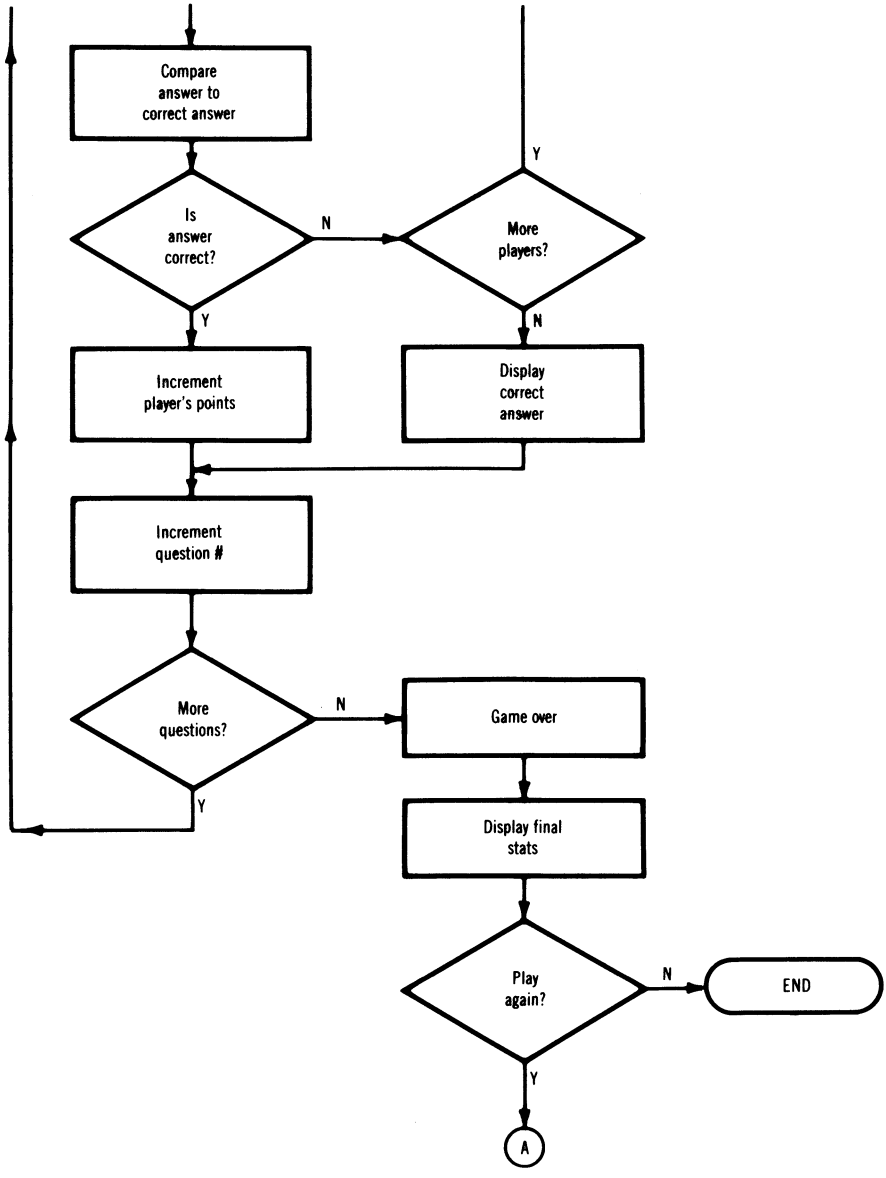
Flowchart 2 cont. — Cassette Game



Flowchart 2 cont. — Cassette Game

(foldout)

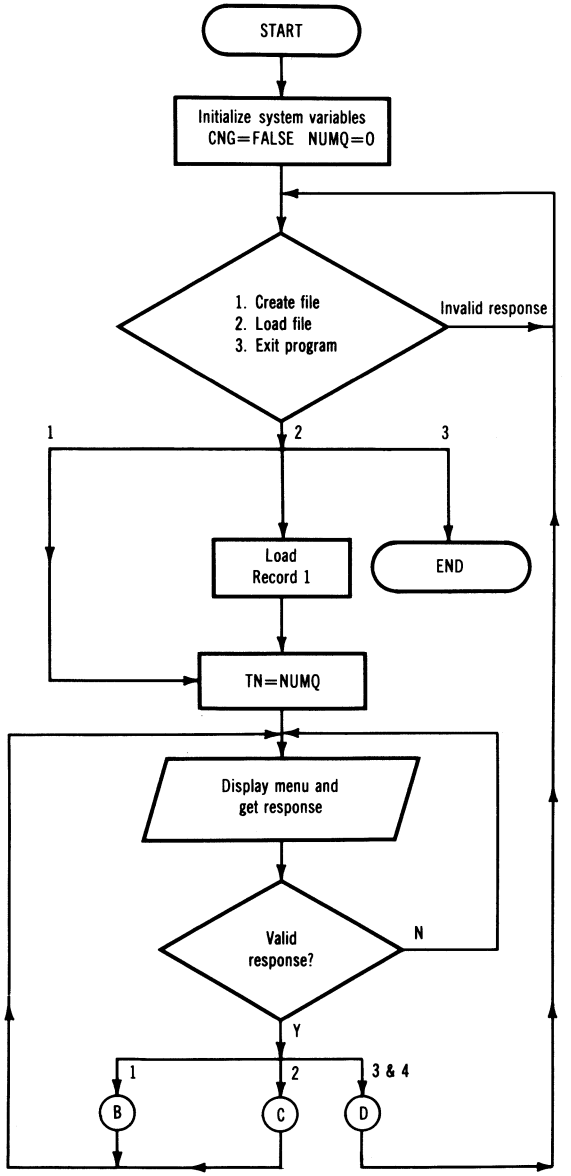
Flowchart 2 cont. — Cassette Game



Flowchart 3 — Disk Data Base

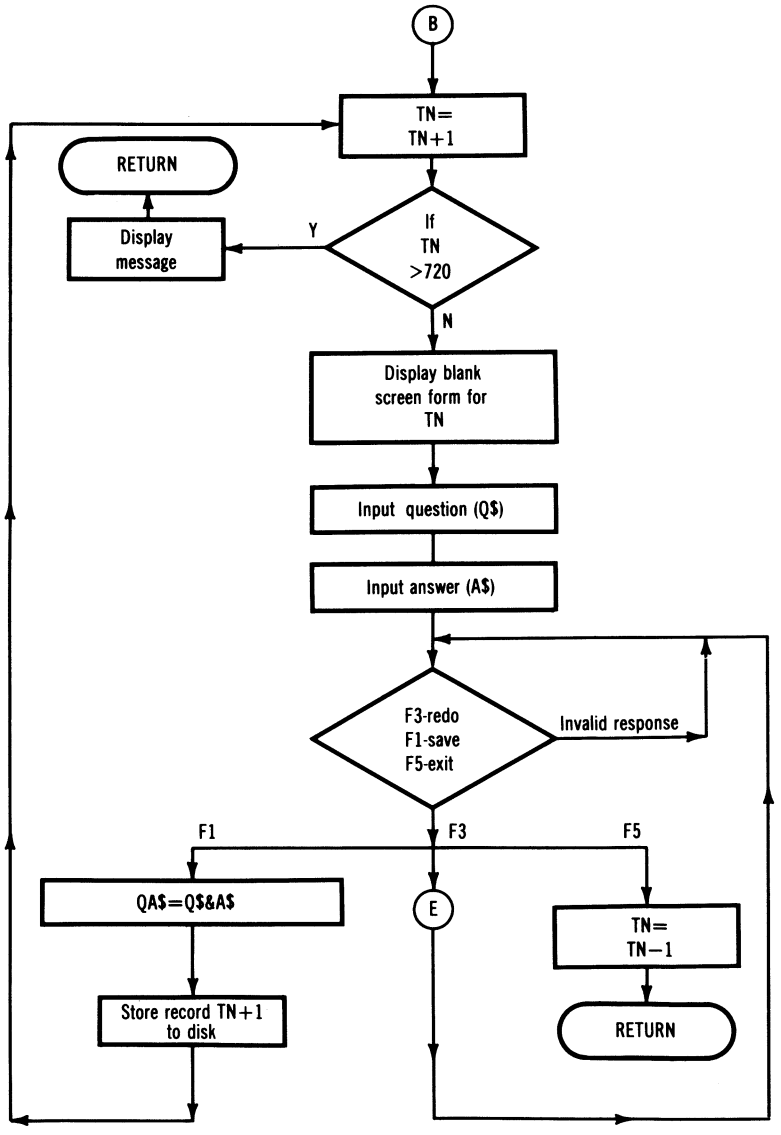

(foldout)

Flowchart 3 — Disk Data Base



Flowchart 3 cont. — Disk Data Base
(foldout)

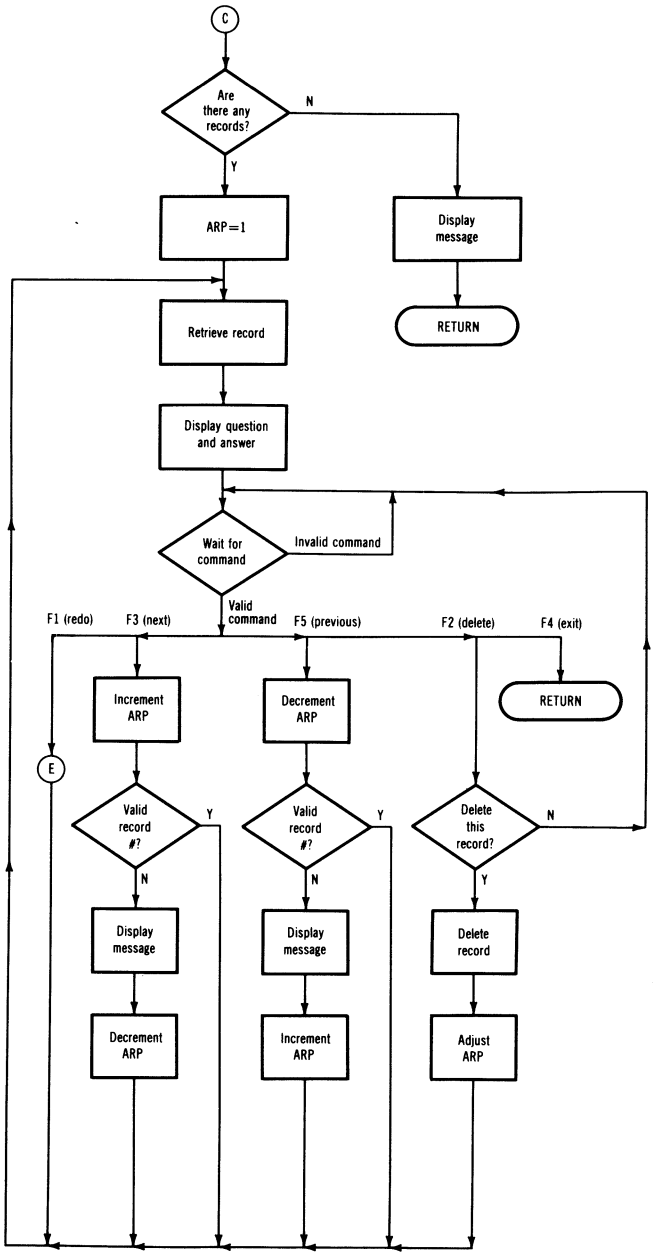
Flowchart 3 cont. — Disk Data Base



Flowchart 3 cont. — Disk Data Base

(foldout)

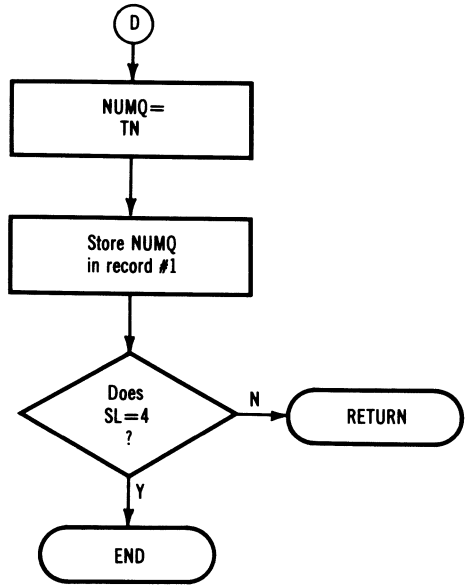
Flowchart 3 cont. — Disk Data Base



Flowchart 3 cont. — Disk Data Base

(foldout)

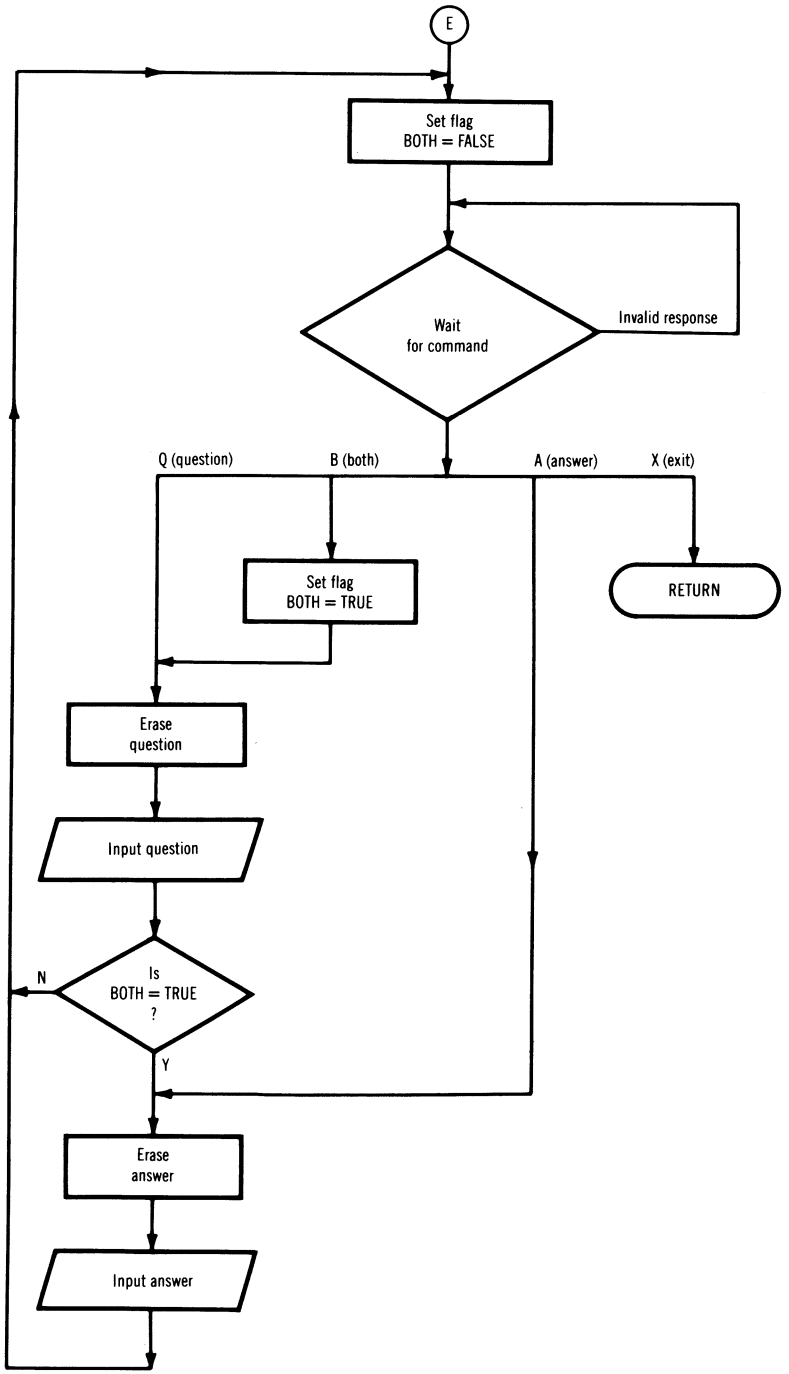
Flowchart 3 cont. — Disk Data Base



Flowchart 3 cont. — Disk Data Base

(foldout)

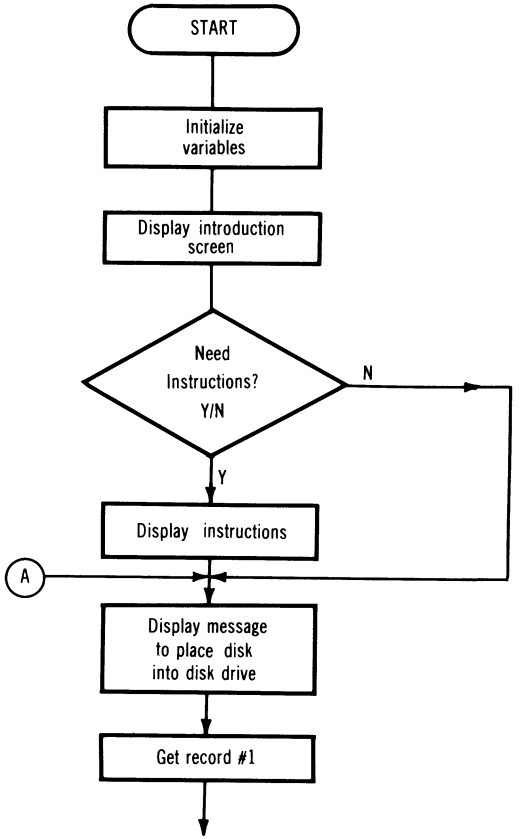
Flowchart 3 cont. — Disk Data Base



Flowchart 4 — Disk Game

(foldout)

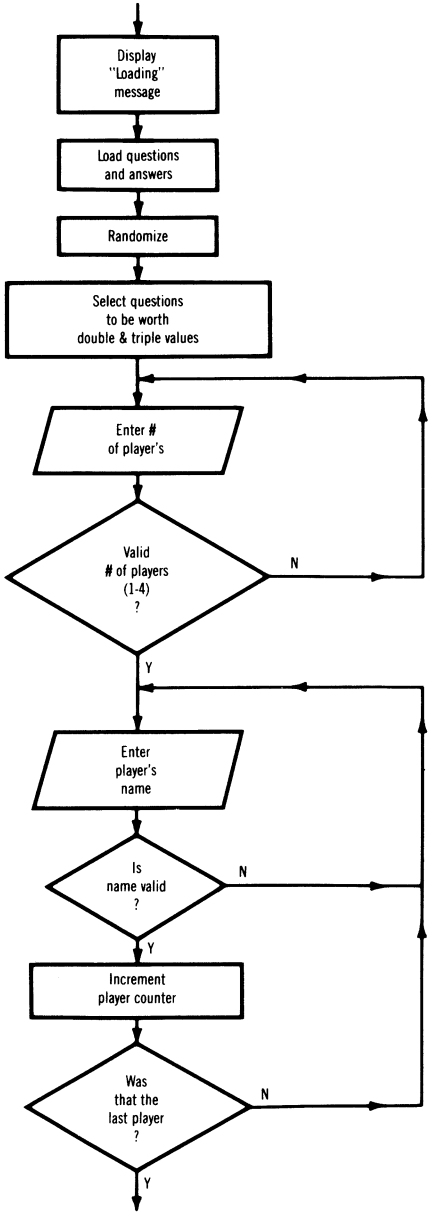
Flowchart 4 — Disk Game



Flowchart 4 cont. — Disk Game

(foldout)

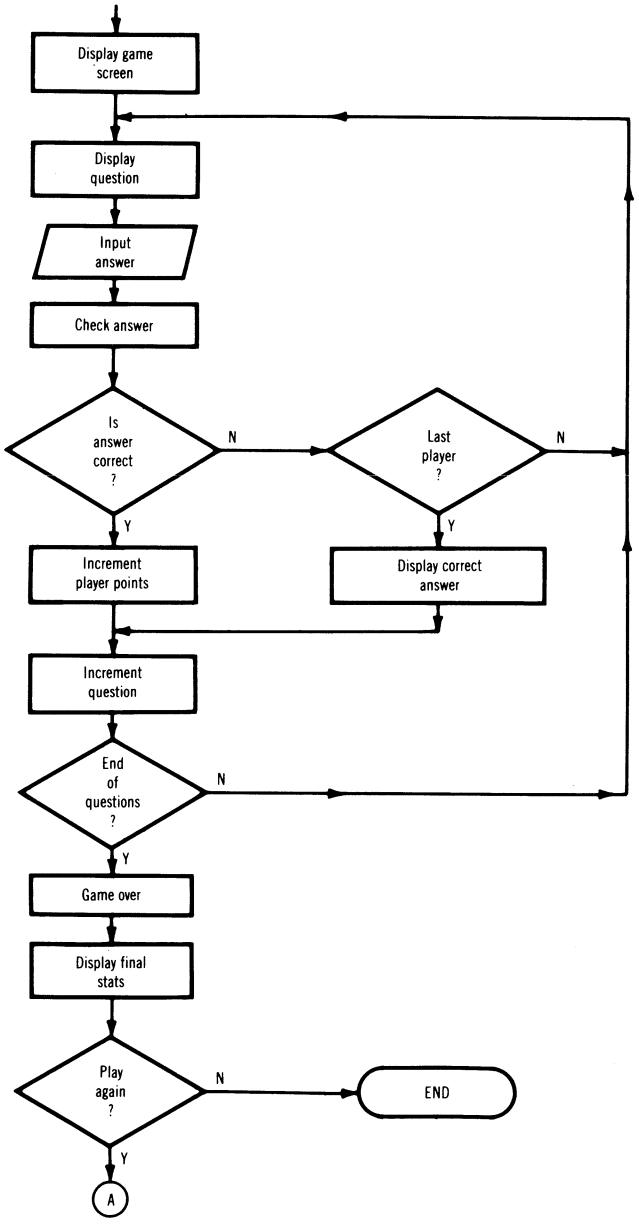
Flowchart 4 cont. — Disk Game



Flowchart 4 cont. — Disk Game

(foldout)

Flowchart 4 cont. — Disk Game



SAMS™

Book Marketing Book

Sams Books cover a wide range of technical topics. We are always looking for more information from you, our readers, as to which additional topics need coverage. Please fill out this questionnaire and return it to us with your suggestions. They will be appreciated.

Please check the areas of interest:

1. CURRENT TECHNOLOGIES
   - ☐ Electronics
   - ☐ Circuit Design
   - ☐ Computers
     - ☐ Business Applications
     - ☐ Fundamentals
     - ☐ Languages _____ Specify _____
     - ☐ Machine Specific:_____ _____ _____
   - ☐ Microprocessors
   - ☐ Networking
   - ☐ Servicing/Repair
   - ☐ _____ Other

2. NEW TECHNOLOGIES
   - ☐ Fiber Optics
   - ☐ Robotics
   - ☐ Security Electronics
   - ☐ Speech Synthesis
   - ☐ Telecommunications
     - ☐ Cellular
     - ☐ Satellite
   - ☐ Video
   - ☐ Other _____ _____ _____ _____

3. Do you ☐ own ☐ operate a personal computer? Model _____

4. Have you bought other Sams Books? Please list: _____ _____ _____

5. OCCUPATION
   - ☐ Business Professional _____ Specify _____
   - ☐ Educator
   - ☐ Engineer _____ Specify
   - ☐ Hobbyist
   - ☐ Programmer
   - ☐ Retailer
   - ☐ Student
   - ☐ _____ Other

6. EDUCATION
   - ☐ High School Graduate
   - ☐ Tech School Graduate
   - ☐ College Graduate
   - ☐ Post-graduate degree

COMMENTS _____ _____ _____

(OPTIONAL)

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

SAMS™

22037 22176 22177 22187 22311 22340 22343 22365 22396

SAMS™

Book Mark

SAMS™

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 1076 INDIANAPOLIS, IND.

POSTAGE WILL BE PAID BY ADDRESSEE

HOWARD W. SAMS & CO., INC.
4300 WEST 62ND STREET
P.O. Box 7092
Indianapolis, IN 46206

ATTENTION: Public Relations Department

SAMS

Commodore 64™ Trivia Data Base

- Introduces Commodore 64 owners to data bases and their applications by enabling users to understand how a simplified data base is created and why it works
- Discusses user-friendly data entry, error checking and program continuity
- Compares advantages and disadvantages of cassette and disk files
- Includes a sample trivia data base file with 100 questions and answers
- Provides tips on making programs run faster and more efficiently
- Helps users put parts of the data base together and learn how to use the program

Commodore 64™ Trivia DataBase is an entertaining way to see for yourself how data bases can be used for fun and education

Machine Requirements:

- **Commodore 64 Computer**
- **Monitor**
- **Disk Drive and/or Datasette**

Howard W. Sams & Co., Inc.
4300 West 62nd Street, Indianapolis, Indiana 46268 U.S.A.

$8.95/22396

ISBN: 0-672-22396-1

Commodore 64 Trivia Data Base

26484

by James F. Hunter and Gregory L. Guntle

Commodore 64

Howard W. Sams & Co., Inc.
4300 West 62nd Street, Indianapolis, Indiana 46268 U.S.A.