APPLE DISK LIBRARIAN

Apple Disk Librarian

by George D. Parker
1702 West Taylor Drive
Carbondale, IL 62901

In the beginning, there was a brand new Apple][with three diskettes: one for games, one for utilities and completed programs, and one for program development. It was very easy to locate a particular program; but slowly, and then not so slowly, the number of disks increased as games were purchased, programs were written or entered from magazine listings, and public domain programs were traded with friends. Soon my memory started to fail as to where a particular program was. What I needed was a master catalog that would help me keep track of all my files. I wrote a simple BASIC program and laboriously entered the names and volumes of over a hundred files, but it was not sufficient.

A friend showed me a multi-disk cataloging program based upon FILE CABINET. It worked, but was very slow reading catalogs from disks as well as reading the master catalog in from disk. Surely I could improve upon that! Of course, I might have to learn assembly language along the way, but the best way I know to learn a new programming language is to write a major program. Besides, I had just bought a book on assembly language by Roger Wagner that needed application. The result was the following multi-disk cataloging system.

The system consists of four major parts: **LIBRARIAN CATALOGER**, which is a BASIC program that creates the initial data base from the catalogs of several disks; **DLC.OBJ**, which is a binary program that does the actual accessing of the catalogs using RWTS routines; **LIBRARIAN FILER**, which is a BASIC program that performs the data management of the master catalog; and **BUBBLE.OBJ**, which is a binary program that performs the sorts requested by LIBRARIAN FILER. Whereas LIBRARIAN CATALOGER is essentially a single purpose program with few options, LIBRARIAN FILER has many useful features: in addition to retrieving and saving the master file, it allows you to sort the file (three ways), print it out (in whole or in part) on a printer or a monitor, search for particular entries, add or delete entries, and edit the entries to include descriptive information or reflect changes in your files.

SYSTEM REQUIREMENTS

This program was developed for a 48K machine. You might modify it to run on a 32K machine but you will lose a lot of data storage. 16K translates into about 400 catalog entries. (See ways to save space in the customizing section below.) A printer is useful to get full value from the system but is not necessary. (In fact, this program was for the most part written before I got a printer. That's fun with programs this size!) An Apple Silentype is adequate, but a more advanced printer which handles pages and form feeds is better.

The system works only with DOS 3.3 disks. If you want to include entries from DOS 3.2 disks, you will have to either enter them by hand using the Add Entries option of LIBRARIAN FILER, or make major modifications to the program.

ENTERING THE PROGRAMS

I keep the entire system on a disk all by itself. **Listing 1** gives the program LIBRARIAN CATALOGER and **Listing 2** gives LIBRARIAN FILER. These two BASIC programs should be entered as is and saved under the names **LIBRARIAN CATALOGER, and LIBRARIAN FILER.** Once you are sure the programs are correctly entered and saved, LOCK them to save grief later.

The two binary programs may be entered either by entering the monitor and typing in their hex codes (on the left side of the listings) or by using an assembler. They were prepared using the Editor/Assembler from the Apple DOS Tool Kit. The only codes that are not 6502 opcodes are the necessary **ORG** and an **ASC** code in **line 175** of DLC.OBJ. This code is used to enter some ASCII characters. You could also use a HEX pseudo-op to enter this data, or do it from the monitor.

If you enter the hex codes for these two programs (or if you add the data in DLC.OBJ from the monitor) you will need to save them to disk. The correct save commands are

```
BSAVE DLC.OBJ,A$300,L$C3
```

and

```
BSAVE BUBBLE.OBJ,A$9900,L$155
```

Once you know the files are successfully saved, LOCK them.

RUNNING DISK LIBRARIAN CATALOGER

A major feature of the system is that most requests for input expect a single keystroke for a reply. This is indicated by choices given in brackets < >. **Resist the temptation to press the RETURN key!** If you don't, you may get temporarily side-tracked as the error trapping routines take over. You may even get sent back to the main menu.

To start, RUN the LIBRARIAN CATALOGER program, and choose option 1. At the beginning of the program you will be given an opportunity to receive some brief instructions and warnings. After that the main menu will appear.

The first option reads catalogs from disks and into memory. You will be asked how many drives you have. **It is assumed that the system disk is in Drive 1 of Slot 6.** If you have two drives, the disks to be cataloged will go in Drive 2. The volume number on the disk being cataloged is read and you are given the option to change this to **any number less than 999.** This allows you to assign volume numbers to your disks without initializing them with that number. The actual number recorded on the disk is not altered.

Normally you will want to read the entire catalog into memory. However, you will have the option to request prompting. This will allow you to choose, one at a time, which file names to add to your master catalog. For example, you can omit all of the text files this way, or binary routines associated with BASIC programs. Note that control characters in filenames are converted to regular upper case letters as they are read in. This is necessary so that they may be edited.

You may enter as many disk catalogs as you want, one after the other, as long as **you don't exceed 500 entries.** This limit could be increased somewhat, but the size of LIBRARIAN FILER limits you to about 575 entries. Also, it is safer to deal with smaller size catalogs and save to disk more frequently in the unlikely event that a power failure causes you to lose some of your work. When you call up Option one a second time while running CATALOGER, the data from the first call is lost. This is done to prevent you from saving duplicate information.

Option two allows you to save the combined catalogs to disk. This data may be put in a new file or appended to the end of an existing file (as long as the total number of file names in the saved file is under 500). If needed, you may list the catalog of the system disk to determine the names of existing master catalog files. I use separate files for games and utilities.

The various error traps are designed to keep you from exiting the program by mistake. The only way to exit the program other than by choosing the appropriate menu option is with a RESET. If you should accidentally exit other than by the menu, you may re-enter without loss of data with the command **GOTO 10000.**

The program's exit options are self-explanatory. If they are used, there is a change in MAXFILES so that data is lost, and the program should not be re-entered at line 10000.

RUNNING DISK LIBRARIAN FILER (DLF)

The cautions regarding single keystroke entries in CATALOGER apply here as well. Likewise, if you should accidentally exit the program, **GOTO 10000** will re-enter.

In Option one you read in the master catalog file. Several files may be combined, as long as the total length stays under 500 entries. Actually, you do not really want to work with a file this big — it takes too long to read in, sort, search, or save. Unlike CATALOGER, this option may be used several times in a session without loss of data. You may read in a file, sort and edit it, and then read in another file which will be appended to it. While using this option, you are able to list the catalog of the system disk to check names of master catalog files.

The second option is straightforward and allows you to save your edited and sorted file to disk. You will have the option to save it under the name of the last file accessed, or under a different name. The system disk can also be checked for existing master catalog files as in option one.

Option three is the important **sort** routine. You may sort your master catalog by filename, filetype, or volume number. If you sort by filename and then by filetype, your catalog will be alphabetized within filetypes. Since it

continued on next page

is often desirable to sort by filenames and break ties by volume number (i.e., sort by volume and then by filename), a special option is available to do this without a double pass. I use this sort for my games and keep utilities sorted by volume.

The sort routine is performed by a binary program (**BUBBLE.OBJ**) which is a bubble-tag sort. **It is relatively fast and can reverse a file of 400 entries in about 45 seconds.** Since you normally first sort a small file and then sort again later as additional entries are made, your sorting times will probably be much shorter. The routine is designed so that entries added to a sorted file "bubble into place" very quickly.

The fourth option allows you to print your sorted and edited catalog. You may send this output either to the screen or to your printer. If you choose screen output, the filenames will be shortened somewhat in order to fit. You will be able to abort the printing at any stage in this sub-option. If you choose to use a printer, the program will ask you to choose either the Apple Silentype or the Epson. Feel free to change the latter to agree with your printer. The main distinction between the two is the Epson's ability to use fan fold paper and do form feeds. In order for these form feeds to work correctly, you must establish top-of-form when you turn on the printer. You can print out the entire catalog, a single filetype, or a single volume.

Option five is the search option. In order to locate a particular entry you need only know the first few letters of the filename. For example, you can list all the entries which begin with the letter 'M'. This is useful when you cannot remember the actual name of the file you want to locate.

The sixth option allows you to add entries to the master catalog. For example, I added the entry **Vol: 14, Name: Alien Rain, Type: B?, Length: ???** because the protected disk could not be cataloged. This option can also be used if you add just a few files to a disk that has already been included in your master file.

Option seven is the reverse of option six—it allows you to delete items. These might be duplicate entries, programs that no longer exist, or all of the HELLO programs. You can delete the entries by name or by sequence number, if you know it. (The search routine reports this sequence number.) Only partial filenames need be given. **If you intend to delete several entries by sequence number, do it in reverse order (high to low number) since deletions change the sequence numbers.** In any case, the item to be deleted is displayed before the actual deletion occurs so that you have one last chance to change your mind.

The eighth option allows you to edit entries. Thus you can change filenames if needed, or change the volume number if you move the program to a different disk. You can add descriptive information to the filename. For example, you might change 'MDC' to 'MDC (MULTI DISK CATALOGER)'. Or you might classify your games as Adventure, Fantasy, Arcade, Space, etc.

The editing is done using **ESC I, J, K, L** and the other standard editing commands. The current entry is displayed exactly as it appears in memory (as a 38 byte character string). You change it just as you would change a line in an Applesoft program.

You can either edit all entries whose filename starts with a particular character string or you can scan the entire file. In the scan mode you can edit an entry, delete it, skip over it, or exit the edit mode. This is especially useful when you first create the master file and want to clean it up a bit.

PROGRAMMING FEATURES IN LIBRARIAN CATALOGER

It seemed to me that part of the problem with the cataloging program was the inefficient handling of strings. I decided right away to keep a single character string for each entry (rather than several), and treat it as a random access file. Thus a 38 byte string was created for each entry — 3 bytes for the volume, 30 for the filename, two for the filetype, and 3 for the sector length. I chose to use the entire 30 character filename so that there would be room to add descriptive information.

Clearly, a binary program is needed to access the catalog on a disk and read it into memory. This is done one sector at a time to limit the amount of temporary storage used. The sector is scanned for data and that data is passed on to the Applesoft program. There are at least two ways to pass data back and forth between binary programs and BASIC programs. I used one method here and another in FILER.

At the very beginning of the program **(line 130)** two variables are declared: **NAME$** and **LN%**. These will appear first in the variable table, and can be found very easily since the address of the beginning of the variable table is stored in bytes **105** and **106** (that is, at Hex **$69** and **$6A**).

A character string variable appears in the variable table as a two byte name, a one byte length, a two byte address of the actual string, and two garbage bytes. The binary program will adjust this information so that **NAME$** will point to a 32 byte string (in the input buffer for RWTS) that contains a filename plus filetype. We adjust the length of **NAME$** in **line 210** to **32** to prepare for this.

An integer variable, such as **LN%** is stored in the variable table as a two byte name, a two byte value (hi-byte first!), and three garbage bytes. **LN%** has its value loaded by DLC.OBJ.

There are three entry points to **DLC.OBJ** and these are defined in line 220 in order to make it easier if DLC.OBJ is modified. (This technique is a good idea in general — it helps to remind you what the calls are for.) These entry points are called **CALRW**, **LOOK**, and **LP** here and **CALRW**, **LOOK**, and **LOOPER** in the assembly listing for DLC.OBJ.

The actual reading of catalogs begins at **line 1090** where RWTS is loaded with the proper parameters: **Track 17, Sector 0, Drive DR%, Slot 6 (note 96 = 6*16), and the read command 1**. Then a call to **CALRW** reads in the first sector of the catalog. This will tell us the volume number and where the next sector of the catalog is located. The latter information is passed to RWTS by the binary program. The volume, or your alternative choice, is stored in the string **VOL$**.

A call to **LOOK** reads the first seven catalog entries into a buffer and processes them one at a time at **line 1210**. The sector length is passed to the integer variable **LN%**. The filetype is determined and placed at the end of the filename. (Note that I place the lock symbol after the filetype.) Finally, the address of this string is put into the variable table for **NAME$**.

Applesoft takes over and checks a flag in **line 1220** to see if the entire catalog has been read. If not, it assigns a value to **PR$(I)** (the I'th entry of the master catalog) based on the values of **NAME$, VOL$**, and **LN%** in line **1230**. **LP**, the third entry point to DLC.OBJ, is used to go after succeeding entries at line **1250**. LP is able to determine whether it is time to read in a new catalog sector.

In order for FILER to correctly read in the master file, it is necessary to store the number of entries at the beginning of the text file. Since we want to be able to append additional entries to this file and change the number of records, it is necessary that this piece of information occupy a fixed number of bytes. I chose to store it as a five byte character string. This string is created in **line 2330**.

Note the use of the strings **OP$, WR$**, and others in **lines 2350 to 2400**. They are defined in **line 2020**. I find it easier when accessing text files to shorten the commands in this manner.

The only other important features of this program appear at its beginning. **MAXFILES** is set at **1** since only one file will ever be accessed by DOS at any one time. This gives an extra 1K of storage. Secondly, **DLC.OBJ** sets the buffer for RWTS at **$9900**, just below the DOS buffer. Thus **HIMEM:** is set at **39167** to protect this buffer from string variables.

HOW DOES DLC.OBJ WORK?

In order to understand this routine, you need to know how the catalog of a standard disk is stored. I learned this from the book BENEATH APPLE DOS by Worth and Lechner. The necessary information can also be found in your DOS manual, although it is not quite as comprehensive. You should also be familiar with RWTS.

I use four zero page variables: one passes the RWTS error code, one is the Applesoft variable table pointer, one is a flag to indicate the end of the catalog, and the most important one, OFFSET, is used while scanning a sector of the catalog. The regular IOB table for RWTS is used.

The first call to RWTS (at **CALRW, LINE 39**) reads in sector zero of the catalog. As long as there is no error (**line 48**) the track and sector of the next catalog sector are determined and stored in **lines 55-59**.

The first actual catalog entry begins in byte **$B** of the next sector. A call to **LOOK** initializes **OFFSET** to **$B** and the first sector of real data is read in by a call to **CALRW**. Normally, bytes **OFFSET** and **OFFSET+1** contain the track and sector of the track/sector list for the file. The entry at **OFFSET** is stored in the completion flag. If this value is **0**, there are no more entries in the catalog. If it is **$FF**, the entry is a deleted file and is skipped with a jump to **LOOPER**. Otherwise, we can start to scan the remaining **33 bytes** that make up the entry.

The file length is stored in the last two bytes of the catalog entry, although the hi-byte is usually left blank because DOS does not fill it. **Lines 83-94** locate the length and transfer it to **LN%**. NOTE THE COMMAND **'CLD' IN LINE 85**. It is crucial that addition be binary and not binary coded decimal. Since this routine is called up by a BASIC program that uses BCD addition, it is necessary to reset this flag to get correct addition. Also note that the sector length is stored lo-byte first in the catalog but must be stored hi-byte first in **LN%**.

Next we go after the filetype which is stored at byte **OFFSET+2**. This must be decoded to

continued on next page

obtain the lock code and the filetype. (The lock code is stored in bit 7 while the filetype is indicated by setting one of the remaining bits.) This is done in lines 102-128. The two byte decoded filetype is stored at the end of the filename where the length had been stored. Note the use of a table to contain the filetype codes. Type R is for relocatable binary files created by the relocating loader in DOS Tool Kit. Type S has been reserved by Apple for future expansion.

The next to the last step is to scan the filename, which starts at byte OFFSET+3 of the entry, and convert control characters to regular characters. Any character whose ASCII code is less than $20 is a control character. We add $40 to it to make the conversion. We also make sure that bit-7 of every character is off since that is the way that Applesoft stores strings: This is done at line 122.

The final step is to pass the address of the filename to the pointer for NAME$ in lines 152-159. This address is $9900 + OFFSET + $3.

Later calls to this routine enter at LOOPER, where OFFSET is incremented by 35 ($23). If seven entries have been read then OFFSET becomes 0 and it is time to read a new sector. A jump to LOOK is made.

THE FEATURES OF DISK LIBRARIAN FILER

Just as with the program CATALOGER, we set MAXFILES equal to 1 to get an extra 1K of storage. HIMEM: is set to 39167 so that the binary program BUBBLE.OBJ can be safely loaded at $9900. Page three of RAM is used for the error routine from the Applesoft Reference Manual, page 82.

One of the problems with using strings in Applesoft is that if they are changed very much, storage is wasted and eventually time has to be spent cleaning up the garbage. I decided to minimize this problem by insulating the strings with a pointer array. Deletions and sorts in the master file are accomplished by working with the pointers instead of the actual strings.

The major variables are defined as follows. P$ is an array of character strings that holds the actual catalog entries. It has dimension 500. P% is an array of integers that point to the entries of the catalog. For example, the seventh sequential catalog entry is given by P$(P%(7)). P% has dimension 500 also. N is the number of entries of P$ that have been used. NUM is the number of actual catalog entries, so that NUM < = N.

Despite recommendations to the contrary, I often find I get less confused if I do not use the zero entry of an array. That is the case here. Neither P%(0) nor P$(0) are used for data storage. However, because of the way that these arrays are passed to BUBBLE.OBJ, P$(0) is initialized to 38 blanks and P%(0) = 0. During sorts P$(0) will always be first. Note the way the P$(0) is initialized in line 140. When a string is initialized in a program line, the pointer in the variable table points to that part of the program, rather than to high memory. This doesn't hurt, but it got me very confused while debugging BUBBLE.OBJ until I avoided the problem with the introduction of BL$.

When the master file is read in from disk, P%(I) is set equal to I. Later the values in P% change. When an entry is added to the catalog, both N and NUM increase since it is placed at the current end of the string array P$. When an entry is deleted, the values of P% are changed to reflect this and NUM is decreased. P$ and N are not affected. During the sorts, the pointers in P% are changed instead of the actual strings. When the file is resaved on disk the pointers allow it to be saved in sorted form.

PROGRAM TECHNIQUES

Some of my favorite programming techniques show up in this program. For example, I like the major portions of a program to start at nice line numbers. This is illustrated by the branch at the main menu in line 320. During program development this makes it easy to remember where various segments are located.

Another thing I like to do is display a flashing message while the machine is performing a major task with no output. This keeps my mind at ease: it looks like something is happening. An example of this is given by line 1100. Once the task is completed I make sure the message is erased by overwriting it. VTAB is useful in this.

Since some tasks take a while, I like to have my attention grabbed when they are down. Likewise, I like error messages brought to my attention, forcefully. Thus you will find a bit of bell ringing in this program. The bells can be turned off by not initializing the variable B$ in line 130.

Options one and two operate in a very straightforward manner and should be easy to understand.

The sort option has most of its work done by BUBBLE.OBJ. All that is needed is to pass to that program several parameters. FLAG is used to indicate if the special double sort is to be made. OFFSET indicates the point within the catalog entry where the sort field begins. NCHRS is the number of characters in the sort field. Finally, NUM, the number of catalog entries, is passed in.

We passed variables between CATALOGER and DLC.OBJ by exploiting the variable table and defining the variables needed first. In this program, we use another method. The names of the arrays are passed to BUBBLE.OBJ as part of the call statement in line 3230. Then BUBBLE.OBJ will locate the arrays using an Applesoft routine, just the way the Applesoft interpreter does it when running a program. (I learned this trick from the program 'Speed Sort' in Nibble, Vol. 3, No. 2.)

The main job of the print routine is to tear the catalog entries apart into several strings just before printing. Line 4320 illustrates this. When printing only a partial catalog, the appropriate field in the entry is tested before printing.

I thought it would be useful when printing to the screen to have a border. This is done using the PRINTs and POKEs in lines 4240-4270 to change the window size. As the pages of the catalog scroll by on the screen, the last entry of one page becomes the first entry of the next because of the re-initialization of K=1 in line 4350. You may avoid this feature by re-initializing K to 0, or get a larger overlap with K = 2 or 3.

When using the Epson printer option, it is assumed that your interface accepts the command ctrl-I60P. This causes the printer to skip six lines after it has printed 60, thus going over the perforation on fan fold paper. If you are not able to use this command you may want to build in a counter right after line 4530 and issue a form feed when the counter reaches 60.

The locate routine is very simple and just scans through the file looking for filenames beginning with a particular character string. Similarly the add entries routine is simple. New entries are placed at the end of the file. The sort routine is designed to move them into correct position very quickly.

In the delete routine the only interesting technique is the variable C in lines 7160, 7180, and 7220. Since deletion by name requires a loop through the entire file, once a file is deleted we need a way back up the loop counter so that we won't skip the next item. If entry 1 is deleted, then entry 2 becomes entry 1 and would be skipped if we went on to look at entry 2.

The same idea is used in the editing option when scanning the entire file and performing some deletions. The edit option also uses the window command POKE 34,3 to protect the top lines of the screen from scrolling. Actual editing of an entry is done by printing the current entry and then issuing an INPUT request. By using the standard editing commands you can manipulate the entry as desired.

WHAT GOES ON IN BUBBLE.OBJ?

This can be the hardest part of the system to understand because of the extensive indirect addressing and the use of P% as a tag.

If we ignore the fact that we actually are going to sort on fields within the entries P$(I), this program is really just an assembly language implementation of the following simple BASIC program.

```
 1 FOR I = 1 TO NUM
 2 FLAG2 = 0
 3 FOR J = NUM TO I STEP -1
 4 IF P$(P%(J-1)) < = P$(P%(J)) THEN 9
 5 FLAG2 = 1
 6 TEMP = P%(J)
 7 P%(J) = P%(J-1)
 8 P%(J-1) = TEMP
 9 NEXT J
10 IF FLAG2 = 0 THEN I = NUM : REM
   DONE, FORCE EXIT
11 NEXT I
```

This is a standard bubble-tag sort. The 'tag' refers to the use of P%. FLAG2 is used to detect when the sort is completed — if we pass through the entire array without setting FLAG2, then the array is sorted. We run through the array backwards so that when new items are added to a sorted array (at the bottom), they will be bubbled into place very quickly. If 5 items are added, it will take a maximum of 6 passes (values of I) to resort. If we sorted from the top it would take a lot longer in general.

I visualize the catalog file as a vertical column of words which I want to be alphabetized. At each stage I must compare two neighboring entries, the upper and the lower. This terminology is reflected in the variable names.

The primary variables are USPTR and LSPTR, which hold the addresses of the pointers to the upper and lower strings; UTGPTR and LTGPTR, which hold the addresses of the upper and lower tags (P%); UTAG and LTAG, which hold the actual values of P%; MAINCT, which corresponds to I in the simple program above; and CTR, which corresponds to J.

The Applesoft routine FNDVAR is used in lines 57 and 65 to locate the beginning of the arrays P$ and P% in the variable table. Since

continued on next page

Apple Disk Librarian (Cont.)

we will pass through the array P% backwards, we calculate the address of the last entry of P% in lines 78-86.

After this it is a question of loading the tag pointers with the appropriate values so that the tags can be evaluated and the string pointers located. This occurs in several steps beginning at line 94. We need to recall that the entries of P% take up two bytes each, while those of P$ take three (one for length and two for an address). That is the cause of the fancy nested loops in lines 130-146.

Once the actual strings have been located, it is a simple process to compare NCHRS bytes starting at the byte OFFSET. The comparison continues until two characters do not match or the end of the field is reached. If the strings are out of order, the tags are reversed in lines 187-196. Then it is easy to obtain the values of the next pair of tag pointers since the tags are stored consecutively in P%. This is done in lines 213-221.

The remaining sections of the program keep the various counters under control. At the very end we call $FDED to ring the bell by printing ctrl-G.

CUSTOMIZING THE PROGRAMS

There are a few changes you can make with this system to make it fit with your own preferences. As noted before, you can stop most of the bell ringing by removing the initialization of B$ early in CATALOGER and FILER.

You may want to change the printer routine somewhat to agree with your system. When typing a several page catalog, the program does not put a heading at the top of each page. If you choose not to use the built-in skip over perforation but instead include a line counter, it should be easy to repeat the heading on each page.

You could create some extra storage space by removing the directions from each program. For each 1K of additional storage you can handle about 25 more entries. Likewise, if you use the Line Cruncher program from Vol. 3, No. 1 of NIBBLE, you should be able to gain some additional space. It won't help much to remove the REMs because there are so few. If you do decide to create more storage and change the dimension of the arrays, don't forget to change the value of the variable SIZE which is used in the error traps while saving in CATALOGER and reading in FILER. SIZE is initialized in line 220 of CATALOGER and line 130 of FILER.

A major change would be in the length of the filenames. Some of you may see no need for 30 characters. Obviously, if this is cut back to 20 characters, storage will be increased by about one third. The changes in CATALOGER are easy, and could be accomplished in lines 1230 and 1280 when the entries in PR$ are created. No change in DLC.OBJ would be needed.

However, making this change in FILER would be more involved. There would have to be changes in any routine that displays or creates a catalog entry because fields would have changed. This would include the Print, Locate, Add, Delete, and Edit options. Also in the Edit option, values in an error trap would have to be changed (lines 8380-8390). In the Sort option, the parameters passed to BUBBLE.OBJ would have to be changed to reflect the correct offset and length. BUBBLE.OBJ itself does not depend on the length of the strings.

BUBBLE.OBJ can be used in other programs where you want a bubble-tag sort. It does assume that all strings in the array have the same length and might give erroneous results if this is not true. For example, if the

continued on page 18

ED. Apple Disk Librarian, Fast GOTO, and Pillar Munch are available on diskette for an introductory price of $19.95 + $1.50 shipping/handling ($2.50 Outside the U.S.) from NIBBLE, P.O. Box 325, Lincoln, MA 01773. Offer expires April 30, 1983.

```
1   REM   ************************
2   REM   * LIBRARIAN CATALOGER *
3   REM   * BY GEORGE D. PARKER *
4   REM   * COPYRIGHT (C) 1983  *
5   REM   * BY MICROSPARC, INC. *
6   REM   * LINCOLN, MA. 01773  *
7   REM   ************************
10 NAME$ = "$":LN% = 0: HOME : VTAB 4: PRINT "
       NIBBLE DISK LIBRARIAN": PRINT : PRINT "       MU
    LTI-DISK CATALOGING SYSTEM"
20  VTAB 23: PRINT "** COPYRIGHT 1983 BY MICROSPARC, I
    NC **": VTAB 10: PRINT "CHOOSE:"
30  PRINT : PRINT "  <1>  RUN LIBRARIAN CATALOGER"
40  PRINT "  <2>  RUN LIBRARIAN FILER"
50  PRINT "  <3>  QUIT"
60  PRINT : PRINT "    YOU WANT?  < >"; CHR$ (8); CHR$
    (8);
70  GET A: PRINT A
90  IF A = 2 THEN  PRINT  CHR$ (4);"RUN LIBRARIAN FILE
    R"
100 PRINT  CHR$ (4);"MAXFILES 1"
110 HIMEM: 39167
120 TEXT : HOME
140 REM  ABOVE VARIABLES MUST BE DEFINED FIRST
150 ONERR  GOTO 700
160 VTAB 9: PRINT "       NIBBLE LIBRARIAN CATALOGER"
170 PRINT : PRINT : HTAB 19: PRINT "BY"
180 PRINT : HTAB 12: PRINT "GEORGE D. PARKER"
185 PRINT : PRINT "COPYRIGHT (C) 1983 BY MICRO-SPARC
    INC": PRINT "       ** ALL RIGHTS RESERVED **"
190 DIM PR$(500)
200 A =  PEEK (105) + 256 *  PEEK (106) + 2
210 POKE A,32
220 CALRW = 768:LOOK = 810:LP = 941:B$ =  CHR$ (7):D$ =
     CHR$ (4):Z$ = "< >" +  CHR$ (8) +  CHR$ (8)
230 PRINT D$;"BLOAD MDC.OBJ,D1"
240 VTAB 23: PRINT "DO YOU WANT INSTRUCTIONS? <Y> <N>
    : ";Z$;
250 GET A$: PRINT A$: IF A$ = "Y" THEN  GOSUB 500
260 HOME : VTAB 5: HTAB 11:
270 PRINT "LIBRARIAN  CATALOGER"
280 PRINT : PRINT : PRINT "MENU:"
290 PRINT "    <1> GET A CATALOG"
300 PRINT "    <2> SAVE MASTER CATALOG FILE"
310 PRINT "    <3> EXIT & RUN LIBRARIAN FILER"
320 PRINT "    <4> EXIT CATALOGER"
330 PRINT : PRINT "     CHOICE? ";
340 PRINT Z$;: GET A$: PRINT A$:A =  ASC (A$) - 48
350 IF A < 1 OR A > 4 THEN  PRINT B$;: GOTO 330
360 ON A GOTO 1000,2000,3000,3030
500 HOME : PRINT
510 PRINT "  THIS PROGRAM IS PART ONE OF A MULTI-": PRINT
    "DISK CATALOGING SYSTEM.  IT READS INTO": PRINT "
    MEMORY THE CATALOGS OF SEVERAL DISKS "
520 PRINT "(UP TO 500 ENTRIES AT A TIME) AND CAN": PRINT
    "APPEND THEM TO AN APPROPRIATE TEXT FILE": PRINT
    "WHICH IS ACCESSED BY THE PROGRAM": PRINT "'LIBRA
    RIAN FILER'."
530 PRINT : PRINT "  IN THIS PROGRAM YOU HAVE TWO OPT
    IONS.": PRINT "YOU MAY READ IN AN ENTIRE CATALOG
    OR": PRINT "YOU MAY REQUEST PROMPTING FOR EACH": PRINT
    "ENTRY AS IN THE APPLE UTILITY 'FID'."
540 PRINT : PRINT "  EACH ENTRY IS STORED AS A SINGLE
     ": PRINT "CHARACTER STRING OF LENGTH 38 --": PRINT
    "A 3 DIGIT VOLUME NUMBER, A 30 CHARACTER": PRINT
    "FILENAME, A 2 CHARACTER FILETYPE, AND": PRINT "A
     3 DIGIT SECTOR LENGTH.  CONTROL"
550 PRINT "CHARACTERS IN FILENAMES ARE CONVERTED ": PRINT
    "TO REGULAR CHARACTERS IN THE MASTER": PRINT "CAT
    ALOG."
560 VTAB 24: HTAB 5: PRINT "PUSH RETURN FOR MORE ";Z$
    ;: GET A$: PRINT
570 HOME : PRINT : PRINT "  MOST QUESTIONS REQUIRE A
    SINGLE": PRINT "KEYSTROKE AND NO RETURN TO ANSWER
    . ": PRINT "THESE ARE INDICATED BY CHOICES IN ": PRINT
    "BRACKETS:  <Y> <N>  OR <1> <2> <3>."
580 PRINT "ILLEGAL ENTRIES ARE TRAPPED AND YOU ARE": PRINT
    "PROMPTED TO RESPOND AGAIN."
590 PRINT : PRINT "  IF YOU SHOULD EXIT THE PROGRAM O
    THER": PRINT "THAN BY THE MENU, YOU MAY RE-ENTER
    WITH": PRINT "THE COMMAND ' GOTO 10000 '.  BECAUS
    E ": PRINT "OF THE CHANGE IN MAXFILES, THIS WILL"
    : PRINT "NOT WORK AFTER A NORMAL EXIT."
600 VTAB 23: PRINT "PRESS RETURN TO CONTINUE ";Z$;: GET
    A$: PRINT
610 RETURN
700 PRINT : PRINT D$;"CLOSE":Y =  PEEK (222): PRINT
710 IF Y = 4 THEN  PRINT B$;"DISK IS WRITE PROTECTED.
    ": GOTO 770
720 IF Y = 9 THEN  PRINT B$;"DISK IS FULL.": GOTO 770
730 IF Y = 10 THEN  PRINT B$;"FILE '";F$;"' IS LOCKED
    .": GOTO 770
740 IF Y = 255 THEN  PRINT B$;"CTRL-C INTERRUPT": GOTO
    780
750 IF Y = 8 THEN  PRINT B$;"I/O ERROR": GOTO 780
760 PRINT B$;"APPLESOFT OR DOS ERROR NUMBER ";Y;" HAS
    ": PRINT "OCCURRED.": PRINT : PRINT "SEE PAGE 200
     OF DOS MANUAL AND PAGE 136": PRINT "OF APPLESOFT
     REFERENCE MANUAL FOR LIST": PRINT "OF ERROR CODE
    S."
770 PRINT : PRINT "REPLACE DISK IN DRIVE 1 IF NEEDED"
    : PRINT : PRINT "PRESS RETURN TO CONTINUE ";Z$;: GET
    A$: PRINT : GOTO 2000
780 PRINT : PRINT "PRESS RETURN TO GO TO MAIN MENU ";
    Z$;: GET A$
790 GOTO 260
1000  HOME : VTAB 3: PRINT "   GETTING A CATALOG": PRINT
    "   ----------------": PRINT
1010  PRINT : PRINT B$;"  THIS OPTION WILL DESTROY ANY
     CATALOG": PRINT "CURRENTLY IN MEMORY (RAM)."
1020  PRINT : PRINT "DO YOU WANT TO:": PRINT "  <S>
     ABORT": PRINT "  <RET>  CONTINUE ": PRINT : PRINT
    "      CHOICE? ";Z$;: GET A$: PRINT A$: IF A$ = "S
    " THEN 260
1030  HOME : VTAB 5
```

continued on page 18

Apple Disk Librarian (Cont.)

strings ALPHA, DELTA, ALPHABET, BETA were stored one right after the other with no blanks padding them out to the same length, they would be alphabetized as ALPHABET, ALPHA, BETA, DELTA because ALPHABET comes before ALPHADEL.

One last major change would be to make the system work with DOS 3.2. Presumably this could be done by adding to DLC.OBJ and having both versions of DOS available for use with a software switch. (Information on accessing both DOS's can be found in Joel Buckley's DOS 3.3/3.2 in Nibble Volume 3 Number 3.) If DLC.OBJ is made very much larger, it will not fit in the free space in page 3. It would have to be put up in high memory along with the alternate DOS.

I hope you enjoy using the Disk Librarian. It has certainly made it a lot easier for me to keep track of hundreds of disk files.

```
1040  PRINT "HOW MANY DRIVES? (1) (2): ";Z$;: GET A$: PRINT
     A$:DR% =  ASC (A$) - 48
1050  IF DR% <  > 1 AND DR% <  > 2 THEN  PRINT : PRINT
     B$;: GOTO 1040
1060 I = 0
1070  PRINT : PRINT : PRINT "INSERT DISK TO BE CATALOG
     ED IN DRIVE ";DR%
1080  PRINT "THEN PRESS RETURN. ";Z$;: GET A$: PRINT
1090  POKE 47084,17: POKE 47085,0: POKE 47082,DR%: POKE
     47081,96: POKE 47092,1
1100  CALL CALRW: REM  READ SECTOR 0
1110  GOSUB 1400
1120 VOL% =  PEEK (39174)
1130  PRINT : PRINT "THIS IS VOLUME ";VOL%
1140  PRINT "TO USE THIS NUMBER PRESS RETURN,": PRINT
     "ELSE ENTER AN ALTERNATIVE NUMBER."
1150  INPUT VOL$: IF VOL$ = "" THEN VOL$ =  STR$ (VOL%
     )
1160  PRINT : PRINT : PRINT "OPTIONS:": PRINT "    <1>
      ENTIRE CATALOG": PRINT "    <2> PROMPTING": PRINT

1170  PRINT "      CHOICE? ";Z$;: GET A$: PRINT A$:A =
      ASC (A$) - 48
1180  IF A <  > 1 AND A <  > 2 THEN  PRINT B$;: GOTO 1
     170
1190  ON A GOTO 1200,1260
1200  PRINT : PRINT : VTAB 23: HTAB 15: FLASH : PRINT
     "CATALOGING";: NORMAL
1210  CALL LOOK: GOSUB 1400: REM   GET FIRST ENTRY OF
     SECTOR
1220  IF  PEEK (255) = 0 THEN 1350
1230 PR$(I) =  RIGHT$ ("00" + VOL$,3) + NAME$ +  RIGHT$
     ("  " +  STR$ (LNX),3)
1240 I = I + 1
1250  CALL LP: GOSUB 1400: GOTO 1220: REM   GET NEXT E
     NTRY OF SECTOR
1260  CALL LOOK: GOSUB 1400: REM   GET FIRST ENTRY OF
     SECTOR
1270  IF  PEEK (255) = 0 THEN 1360
1280 PR$(I) =  RIGHT$ ("00" + VOL$,3) + NAME$ +  RIGHT$
     ("  " +  STR$ (LNX),3)
1290  PRINT "-  -  -  -  -  -  -  -  -  -  -  -  -"
1300  PRINT "   " +  RIGHT$ (PR$(I),35): PRINT
1310  PRINT "      ACCEPT? <Y> <N>: ";Z$;: GET A$: PRINT
     A$
1320  IF A$ = "Y" THEN I = I + 1: GOTO 1340
1330  IF A$ <  > "N" THEN  PRINT B$;: GOTO 1310
1340  CALL LP: GOSUB 1400: GOTO 1270: REM   GET NEXT E
     NTRY OF SECTOR
1350  VTAB 23: HTAB 15: PRINT "                "
1360  PRINT : PRINT : PRINT  CHR$ (7);"  DONE!  FILE N
     OW HAS ";I;" ENTRIES.": PRINT : PRINT "CATALOG AN
     OTHER DISK? <Y> <N>: ";Z$;
1370  GET A$: PRINT A$: IF A$ = "Y" THEN 1070
1380  IF DR% = 2 THEN 260
1390  PRINT : PRINT : PRINT B$;"REPLACE MASTER DISK AN
     D PUSH RETURN ";Z$;: GET A$: PRINT : GOTO 260
1400 ERR =  PEEK (253): IF ERR = 0 THEN  RETURN
1410  IF ERR = 8 THEN  PRINT : PRINT B$;"I/O ERROR": GOTO
     1430
1420  PRINT : PRINT  CHR$ (7);"RWTS ERROR NUMBER ";ERR
     ;" HAS OCCURRED."
1430  PRINT "CANNOT READ ANY MORE FROM THIS DISK": PRINT
     : PRINT "PRESS RETURN FOR MAIN MENU. ";Z$;: GET A
     $: GOTO 260
2000  HOME : VTAB 3
2010  PRINT "   SAVING CATALOG FILE": PRINT "   ------
     -------------": PRINT
2020 OP$ = D$ + "OPEN ":CL$ = D$ + "CLOSE ":RE$ = D$ +
     "READ ":WR$ = D$ + "WRITE "
2030  PRINT : PRINT "WHAT IS NAME OF CATALOG FILE?  (E
     NTER": PRINT : PRINT "$ TO ABORT OR @ TO LIST THE
      CATALOG OF": PRINT : PRINT "YOUR MASTER DISK.)";

2040  INPUT "   ";F$
2050  IF F$ = "$" THEN 260
2060  IF  LEN (F$) = 0 THEN  PRINT B$;: GOTO 2030
2070  IF F$ <  > "@" THEN 2100
2080  PRINT D$;"CATALOG,D1"
2090  PRINT : PRINT "PRESS RETURN TO CONTINUE ";Z$;: GET
     A$: PRINT : GOTO 2010
2100  PRINT : PRINT : PRINT "NEW FILE OR APPEND TO OLD
     ? <N> <O>: ";Z$;: GET A$: PRINT A$: PRINT : PRINT

2110  IF A$ = "N" THEN 2300
2120  IF A$ <  > "O" THEN  PRINT B$;: GOTO 2100
2130  PRINT OP$;F$
2140  PRINT RE$;F$
2150  ONERR  GOTO 2190: REM  SPECIAL ERROR TRAP FOR EM
     PTY FILE
2160  INPUT N$
2170  ONERR  GOTO 700
2180  GOTO 2250
2190  ONERR  GOTO 700
2200  IF  PEEK (222) <  > 5 THEN  GOTO 700
2210  PRINT CL$
2220  PRINT : PRINT B$;"FILE '";F$;"' DOES NOT EXIST."
     : PRINT : PRINT "DO YOU WANT TO USE IT ANYWAY? <Y
     ><N>:": PRINT Z$;: GET A$: PRINT A$
2230  IF A$ <  > "Y" THEN 2030
2240  GOTO 2320
2250  PRINT CL$
2260 N =  VAL (N$)
2270  IF N + I <  = 700 THEN 2330
2280  PRINT : PRINT : PRINT B$;"CANNOT APPEND TO FILE
     '";F$;"'": PRINT "BECAUSE IT WILL HAVE ";N + I;"
     ENTRIES."
2290  PRINT : PRINT "RESTART WITH NEW FILENAME": PRINT
     : PRINT "PRESS RETURN TO CONTINUE. ";Z$;: GET A$:
      GOTO 2000
2300  PRINT OP$;F$
2310  PRINT D$;"DELETE ";F$
2320 N = 0
2330 N = N + I:N$ =  LEFT$ ( STR$ (N) + "    ",5)
2340  VTAB 23: HTAB 16: FLASH : PRINT "WRITING": NORMAL
2350  PRINT OP$;F$
2360  PRINT WR$;F$
2370  PRINT N$
2380  PRINT CL$
2390  PRINT D$;"APPEND ";F$
2400  PRINT WR$;F$
2410  FOR J = 0 TO I - 1
2420  PRINT PR$(J)
2430  NEXT
2440  PRINT CL$
2460  VTAB 23: PRINT B$;"FILE '";F$;"' HAS ";N;" ENTRI
     ES NOW.": PRINT : PRINT "PRESS RETURN TO CONTINUE
      ";Z$;: GET A$: PRINT A$
2470  GOTO 260
3000  GOSUB 3070
3010  PRINT : PRINT D$;"RUN LIBRARIAN FILER"
3020  END
3030  GOSUB 3070
3040  PRINT : PRINT D$;"MAXFILES 3"
3050  PRINT B$;"BYE-BYE";B$
3060  END
3070  PRINT B$: PRINT "BE SURE YOU HAVE SAVED YOUR CAT
     ALOG FILE": PRINT : PRINT "DO YOU REALLY WANT TO
     EXIT? <Y> <N>: ";Z$;: GET A$: PRINT A$
3080  IF A$ <  > "Y" THEN 260
3090  RETURN
10000  GOTO 260: REM  RE-ENTRY VECTOR
```

```
1  REM  ************************
2  REM  * LIBRARIAN FILER      *
3  REM  * BY GEORGE D. PARKER  *
4  REM  * COPYRIGHT (C) 1983   *
5  REM  * BY MICROSPARC, INC.  *
6  REM  * LINCOLN, MA. 01773   *
7  REM  ************************
100  TEXT : HOME : PRINT  CHR$ (4);"MAXFILES 1"
110  HIMEM: 39167
120  VTAB 5: HTAB 9: PRINT "NIBBLE LIBRARIAN FILER": VTAB
     8: HTAB 19: PRINT "BY": VTAB 10: HTAB 12: PRINT "
     GEORGE D. PARKER"
125  PRINT : PRINT "COPYRIGHT (C) 1983 BY MICRO-SPARC,
      INC": PRINT "       ** ALL RIGHTS RESERVED **"
130  DIM P%(700),P$(700):D$ =  CHR$ (4):B$ =  CHR$ (7)
     :NUM = 0:N = 0:Z$ = "< >" +  CHR$ (8) +  CHR$ (8)

140 BL$ = "                                          ":P$(
     0) = BL$:TAB$ =  LEFT$ (BL$,10): REM  BL$ IS 38 S
     PACES
150  PRINT D$;"BLOAD BUBBLE.OBJ,D1"
160  ONERR  GOTO 700
```

continued on next page

Apple Disk Librarian (Cont.)

```
170  POKE 768,104: POKE 769,168: POKE 770,104: POKE 77
     1,166: POKE 772,223: POKE 773,154: POKE 774,72: POKE
     775,152: POKE 776,72: POKE 777,96: REM    ERR ROU
     TINE PAGE 82, APPLESOFT REFERENCE MANUAL
180 DP$ = D$ + "OPEN ":RE$ = D$ + "READ ":CL$ = D$ + "
     CLOSE ":WR$ = D$ + "WRITE ":CA$ = D$ + "CATALOG"
190  VTAB 23: PRINT "DO YOU WANT INSTRUCTIONS? <Y> <N>
     : ";Z$;: GET A$: PRINT A$: IF A$ = "Y" THEN 500
200  HOME : VTAB 3: PRINT "MENU:": PRINT
210  PRINT "  <1>  GET MASTER CATALOG FILE"
220  PRINT "  <2>  SAVE MASTER CATALOG FILE"
230  PRINT "  <3>  SORT MASTER CATALOG"
240  PRINT "  <4>  PRINT MASTER CATALOG"
250  PRINT "  <5>  LOCATE ENTRIES IN CATALOG"
260  PRINT "  <6>  ADD ENTRIES"
270  PRINT "  <7>  DELETE ENTRIES"
280  PRINT "  <8>  EDIT ENTRIES"
290  PRINT "  <9>  EXIT LIBRARIAN FILER"
300  PRINT : PRINT "      CHOICE? ";Z$;: GET A$: PRINT
     A$:A =  ASC (A$) - 48
310  IF A < 1 OR A > 9 THEN  PRINT B$;: GOTO 300
320  ON A GOTO 1000,2000,3000,4000,5000,6000,7000,8000
     ,9000
500  HOME
510  PRINT "   THIS IS THE SECOND PART OF A THE    ": PRINT
     "NIBBLE LIBRARIAN SYSTEM.  WHILE THE": PRINT "PRO
     GRAM 'CATALOGER' CREATES THE INITIAL": PRINT "DAT
     A BASE BY READING CATALOGS, THIS"
520  PRINT "PART DOES THE ACTUAL MANAGEMENT."
530  PRINT : PRINT "   THE PROGRAM IS MENU DRIVEN AND
     ": PRINT "ALLOWS THE USER TO COMBINE DATA BASES,"
     : PRINT "ADD OR DELETE OR EDIT OR SORT ENTRIES,":
      PRINT "AND PRINT OUT THE MASTER CATALOG. THE "
540  PRINT "CATALOG MAY CONTAIN UP TO 700 ENTRIES."
550  PRINT : PRINT "   THE SORT ROUTINE ALLOWS SORTING
      BY": PRINT "VOLUME, FILETYPE, OR FILENAME."
560  PRINT : PRINT "   THE PRINT ROUTINE ALLOWS FOR TW
     O": PRINT "OPTIONS -- SCREEN AND PRINTER.  THE": PRINT
     "PRINTER OPTION MAY NEED TO BE REVISED": PRINT "T
     O WORK WITH YOUR PRINTER."
570  VTAB 23: PRINT "PRESS RETURN TO CONTINUE ";Z$;: GET
     A$: PRINT A$
580  HOME : PRINT "  IF BY MISTAKE YOU SHOULD EXIT THE
      ": PRINT "PROGRAM (BY AN ILLEGAL ENTRY?) YOU MAY
     ": PRINT "RE-ENTER THE PROGRAM WITH 'GOTO 10000'.
     "
590  PRINT : PRINT "  THIS WILL NOT WORK IF YOU MAKE A
     ": PRINT "NORMAL EXIT VIA THE MENU BECAUSE THERE"
     : PRINT "WILL HAVE BEEN A CHANGE IN THE NUMBER": PRINT
     "OF FILE BUFFERS (MAXFILES)."
600  PRINT : PRINT "  MOST ANSWERS TO QUESTIONS REQUIR
     E A": PRINT "SINGLE KEYSTROKE AND NO RETURN.  THI
     S": PRINT "IS INDICATED BY CHOICES IN BRACKETS <>
     .": PRINT "ILLEGAL ENTRIES WILL BE TRAPPED AND YO
     U"
610  PRINT "WILL BE PROMPTED TO RESPOND AGAIN."
620  VTAB 23: PRINT "PRESS RETURN TO CONTINUE ";Z$;: GET
     A$: GOTO 200
700  CALL 768
710  PRINT : PRINT D$;"CLOSE";Y =  PEEK (222)
720  IF Y = 5 THEN  PRINT B$;"PREMATURE END OF DATA ON
      FILE ";F$: PRINT "CANNOT READ MORE.":N = N + I -
     1:NUM = NUM + I - 1: CALL  - 958: GOTO 1070
730  IF Y = 4 THEN  PRINT B$;"DISK IS WRITE PROTECTED.
     ": CALL  - 958: GOTO 800
740  IF Y = 9 THEN  PRINT B$;"DISK IS FULL.": CALL  -
     958: GOTO 800
750  IF Y = 10 THEN  PRINT B$;"FILE IS LOCKED.": CALL
      - 958: GOTO 800
760  IF Y = 8 THEN  PRINT B$;"I/O ERROR.": CALL  - 958
     : GOTO 810
770  IF Y = 255 THEN  PRINT B$;"CTRL-C INTERRUPT": GOTO
     810
780  IF Y = 254 THEN  PRINT B$;"NUMERICAL ENTRY EXPECT
     ED": PRINT : RESUME
790  PRINT B$;"APPLESOFT OR DOS ERROR NUMBER ";Y;" HAS
     ": PRINT "OCCURRED.  SEE PAGE 81 OF APPLESOFT": PRINT
     "REFERENCE MANUAL AND PAGE 200 OF DOS": PRINT "MA
     NUAL FOR LIST OF ERROR CODES.": GOTO 810
800  PRINT : PRINT "REPLACE DISK IN DRIVE 1 IF NEEDED.
     ": PRINT : PRINT "PRESS RETURN TO CONTINUE ";Z$;:
      GET A$: GOTO 2000
810  PRINT : PRINT "PRESS RETURN TO GO TO MAIN MENU ";
     Z$;: GET A$
820  GOTO 200
1000  HOME
1010  PRINT "  GET MASTER CATALOG"
1020  PRINT : PRINT "WHAT IS NAME OF CATALOG FILE? (EN
     TER $": PRINT : PRINT "TO ABORT OR @ TO LIST CATA
     LOG OF MASTER": PRINT : PRINT "DISK.) ";: INPUT F
     $
1030  IF  LEN (F$) = 0 THEN  PRINT B$;: GOTO 1010
1040  IF F$ = "$" GOTO 200
1050  IF F$ <  > "@" THEN 1080
1060  PRINT CA$",D1"
1070  PRINT : PRINT "PRESS RETURN TO CONTINUE ";Z$;: GET
     A$: PRINT : PRINT : PRINT : GOTO 1010
1080  PRINT : PRINT : PRINT : PRINT
1100  VTAB 23: HTAB 17: FLASH : PRINT "READING": NORMAL
     : VTAB 14
1110  PRINT OP$;F$
1120  PRINT RE$;F$
1130  ONERR  GOTO 1170: REM  SPECIAL ERROR TRAP FOR EM
     PTY FILE
1140  INPUT M
1150  ONERR  GOTO 700
1160  GOTO 1210
1170  ONERR  GOTO 700
1180  IF  PEEK (222) <  > 5 GOTO 700
1190  PRINT CL$: CALL  - 958
1200  PRINT B$;"FILE '";F$;"' IS EMPTY.": PRINT : PRINT
     "PRESS RETURN TO CONTINUE ";Z$;: GET A$: PRINT : GOTO
     1020
1210  IF M + N <  = 700 THEN 1250
1220  VTAB 23: PRINT B$;"CANNOT LOAD FILE -- TOO BIG.
      THE FILE": PRINT "IN RAM HAS ";N;" ENTRIES AND T
     HE FILE": PRINT "'"F$;"' HAS ";M;" ENTRIES."
1230  PRINT CL$: PRINT
1240  PRINT "PRESS RETURN TO CONTINUE ";Z$;: GET A$: PRINT
     : GOTO 200
1250  FOR I = 1 TO M: INPUT P$(N + I):P%(NUM + I) = N +
     I: NEXT
1260 N = N + M:NUM = NUM + M: PRINT CL$;F$
1270  VTAB 21: PRINT "THE CATALOG NOW HAS ";NUM;" ENTR
     IES."
1280  VTAB 23: PRINT "MERGE ANOTHER CATALOG? <Y> <N>:
     ";B$;Z$;: GET A$: PRINT A$: IF A$ = "Y" THEN  PRINT
     : PRINT : GOTO 1010
1290  GOTO 200
2000  HOME
2010  PRINT "  SAVE MASTER CATALOG"
2020  PRINT : PRINT "DO YOU WISH TO SORT BEFORE SAVING
     ?": PRINT "  <Y>, <N>, OR <$> TO ABORT: ";Z$;: GET
     A$: PRINT A$
2030  IF A$ = "$" THEN 200
2040  IF A$ = "Y" THEN 3000
2050  IF A$ <  > "N" THEN  PRINT B$;: GOTO 2010
2060  PRINT : PRINT "SAVE IN FILE  '";F$;"' ?": PRINT
     "<Y> <N>: ";Z$;: GET A$: PRINT A$: IF A$ = "Y" THEN
     2120
2070  PRINT : PRINT "WHAT FILENAME? (ENTER @ TO LIST T
     HE": PRINT "CATALOG OF YOUR MASTER DISK) ";: INPUT
     F$
2080  IF  LEN (F$) = 0 THEN  PRINT B$;: GOTO 2070
2090  IF F$ <  > "@" THEN 2120
2100  PRINT CA$;",D1"
2110  PRINT : PRINT "PRESS RETURN TO CONTINUE ";Z$;: GET
     A$: PRINT : PRINT : GOTO 2070
2120 N$ =  LEFT$ ( STR$ (NUM) + "     ",5)
2130  PRINT : VTAB 23: HTAB 16: FLASH : PRINT "WRITING
     ": NORMAL
2140  PRINT OP$;F$: PRINT WR$;F$
2150  PRINT N$
2160  FOR I = 1 TO NUM: PRINT P$(P%(I)): NEXT
2170  PRINT CL$;F$
2180  VTAB 23: PRINT "PRESS RETURN TO CONTINUE  ";B$;Z
     $;: GET A$: GOTO 200
3000  HOME
3010  PRINT "  SORT MASTER CATALOG"
3020  VTAB 4: PRINT "  A STANDARD SORT IS DOUBLE -- PR
     IMARY": PRINT "BY FILENAME AND SECONDARY BY VOLUM
     E."
3030  PRINT : PRINT "YOU MAY MAKE A CUSTOM SORT BY USI
     NG": PRINT "OPTIONS 2,3,AND 4.  IF YOU WANT BOTH
     A": PRINT "PRIMARY AND SECONDARY SORT, DO THE": PRINT
     "SECONDARY SORT FIRST."
3040  PRINT : PRINT "MENU:": PRINT
3050  PRINT "  <1>  STANDARD SORT"
3060  PRINT "  <2>  SORT BY FILENAME"
3070  PRINT "  <3>  SORT BY VOLUME"
3080  PRINT "  <4>  SORT BY FILETYPE (A,B,I,R,T)
3090  PRINT "  <5>  EXIT AND RETURN TO MAIN MENU"
3100  PRINT : PRINT "      CHOICE? ";Z$;: GET A$: PRINT
     A$: PRINT :A =  ASC (A$) - 48
3110  IF A < 1 OR A > 5 THEN  PRINT B$;: GOTO 3100
3120  IF A = 5 THEN 200
3130  VTAB 23: HTAB 17: FLASH : PRINT "SORTING": NORMAL

3140  IF A = 1 THEN OFFSET = 3:NCHRS = 30:FLAG = 3
3150  IF A = 2 THEN OFFSET = 3:NCHRS = 30:FLAG = 0
3160  IF A = 3 THEN OFFSET = 0:NCHRS = 3:FLAG = 0
3170  IF A = 4 THEN OFFSET = 33:NCHRS = 1:FLAG = 0
3180  POKE 963,FLAG
3190  POKE 964,OFFSET
3200  POKE 965,NCHRS
3210  POKE 954,NUM - 256 *  INT (NUM / 256)
3220  POKE 955, INT (NUM / 256)
3230  CALL 39168P$(0),P%(0)
3240  VTAB 23: PRINT B$;" TIME TO MAKE A NEW CHOICE (1
     - 5)": GOTO 3100
4000  HOME
```

continued on next page

Apple Disk Librarian (Cont.)

```
4010  PRINT "  PRINT CATALOG"
4020  PRINT : PRINT "OPTIONS:": PRINT
4030  PRINT "    <1>  SCREEN OUTPUT"
4040  PRINT "    <2>  PRINTER OUTPUT"
4050  PRINT "    <3>  EXIT"
4060  PRINT : PRINT "     CHOICE? ";Z$;: GET A$: PRINT
     A$:B =  ASC (A$) - 48
4070  IF B < 1 OR B > 3 THEN  PRINT B$;: GOTO 4060
4080  IF B <  > 2 THEN 4150
4090  PRINT : PRINT "TYPE OF PRINTER?": PRINT
4100  PRINT "  <1> SILENTYPE"
4110  PRINT "  <2> EPSON"
4120  PRINT : PRINT "     WHICH? ";Z$;: GET A$: PRINT
     A$:PT =  ASC (A$) - 48
4130  IF PT < 1 OR PT > 2 THEN  PRINT B$;: GOTO 4120
4140  IF PT = 2 THEN  PRINT : PRINT B$;"BE SURE TO POS
     ITION PAPER AT THE TOP OF": PRINT "A SHEET WITH T
     HAT DEFINED AS  'TOF'.": PRINT "(TURN PRINTER OFF
      AND ON TO BE SURE.)"
4150  IF B = 3 THEN 200
4160  PRINT : PRINT "OPTIONS:"
4170  PRINT : PRINT "   <1>  ENTIRE CATALOG (AS SORTED
     )"
4180  PRINT "    <2>  SINGLE VOLUME"
4190  PRINT "    <3>  SINGLE FILETYPE"
4200  PRINT : PRINT "     CHOICE? ";Z$;: GET A$: PRINT
     A$:A =  ASC (A$) - 48: IF A < 1 OR A > 3 THEN  PRINT
     B$;: GOTO 4200
4210  IF A = 2 THEN  PRINT : PRINT "WHAT VOL? ";: INPUT
     VOL$:VOL$ =  RIGHT$ ("000" + VOL$,3):FF$ = "N": IF
     (B = 2) AND (PT = 2) THEN  PRINT : PRINT "DO YOU
     WANT A FORM FEED BEFORE PRINTING? <Y> <N>: ";Z$;:
      GET FF$: PRINT FF$
4220  IF A = 3 THEN  PRINT : PRINT "WHAT FILETYPE? <A>
     <B> <I> <R> <S> <T>: ";Z$;: GET T$: PRINT T$
4230  IF B = 2 THEN 4390
4240  HOME : PRINT "VOL  FILENAME (SHORT)      TYPE  LE
     N"
4250  PRINT "-----------------------------------"
4260  VTAB 22: PRINT "-----------------------------------
     ---"
4270  VTAB 23: PRINT "PRESS RETURN FOR MORE OR ESCAPE
     TO STOP";: POKE 34,2: POKE 35,21: VTAB 3: HTAB 1:
     K = 0
4280  FOR I = 1 TO NUM
4290  IF A = 1 THEN 4320
4300  IF A = 2 THEN  IF  LEFT$ (P$(PX(I)),3) <  > VOL$
      GOTO 4360
4310  IF A = 3 THEN  IF  MID$ (P$(PX(I)),34,1) <  > T$
      GOTO 4360
4320  PRINT  LEFT$ (P$(PX(I)),3);"  " MID$ (P$(PX(I)),
     4,20);"  "; MID$ (P$(PX(I)),34,2);"  "; RIGHT$ (P
     $(PX(I)),3);
4330 K = K + 1: IF K <  > 19 THEN  PRINT : GOTO 4360
4340  PRINT " ";B$;Z$;: GET A$: IF A$ =  CHR$ (27) THEN
     I = NUM: GOTO 4360
4350  PRINT :K = 1
4360  NEXT
4370  POKE 34,0: POKE 35,24: VTAB 23: HTAB 1: PRINT B$
     ;"PRESS RETURN FOR PRINT MENU          ";B$;Z$;
4380  GET A$: PRINT : GOTO 4020
4390  PR# 1
4400  IF PT = 2 THEN  PRINT  CHR$ (9);"80N": PRINT  CHR$
     (9);"60P": IF FF$ = "Y" THEN  PRINT  CHR$ (12)
4410  PRINT
4420  PRINT TAB$;"VOL   FILENAME
       TYPE  LENGTH"
4430  PRINT TAB$;"-----------------------------------
     ---------------"
4440  PRINT
4450  FOR I = 1 TO NUM
4460  IF A = 1 THEN 4490
4470  IF A = 2 THEN  IF  LEFT$ (P$(PX(I)),3) <  > VOL$
      GOTO 4500
4480  IF A = 3 THEN  IF  MID$ (P$(PX(I)),34,1) <  > T$
      THEN 4500
4490  GOSUB 4530
4500  NEXT : PRINT  CHR$ (9);"0P"
4510  PR# 0
4520  PRINT : PRINT B$;"PRESS RETURN FOR PRINT MENU ";
     Z$;: GET A$: PRINT : GOTO 4010
4530  PRINT TAB$; LEFT$ (P$(PX(I)),3);"   "; MID$ (P$(
     PX(I)),4,30);"   "; MID$ (P$(PX(I)),34,2);"    ";
      RIGHT$ (P$(PX(I)),3)
4540  RETURN
5000  HOME
5010  PRINT "  LOCATE ENTRIES"
5020  PRINT : PRINT "  THIS SEGMENT WILL LOCATE ALL EN
     TRIES": PRINT "WHOSE FILENAME BEGINS WITH WHAT YO
     U": PRINT "ENTER BELOW."
5030  PRINT : PRINT "ENTER FILENAME  ($ TO EXIT)": INPUT
     NAME$:L =  LEN (NAME$)
5040  IF L = 0 THEN  PRINT B$;: GOTO 5030
5050  IF NAME$ = "$" THEN 200
5060 K = 0: PRINT : PRINT : PRINT : GOSUB 5160
5070  FOR I = 1 TO NUM
5080  IF  MID$ (P$(PX(I)),4,L) <  > NAME$ THEN 5110
5090  GOSUB 5140:K = K + 1: IF K <  > 20 *  INT (K / 2
     0) THEN 5110
5100  PRINT : PRINT "PRESS RETURN FOR MORE ";Z$;: GET
     A$: PRINT : PRINT : GOSUB 5160
5110  NEXT
5120  PRINT : PRINT B$;"   THAT'S ALL"
5130  PRINT : PRINT "PRESS RETURN TO CONTINUE ";Z$;: GET
     A$: GOTO 200
5140  PRINT  RIGHT$ ("  " +  STR$ (I),3);"   "; LEFT$
     (P$(PX(I)),3);"   "; MID$ (P$(PX(I)),4,20);"  "; MID$
     (P$(PX(I)),34,2)
5150  RETURN
5160  PRINT "ENTRY VOL  FILENAME (SHORT)      TYPE"
5170  RETURN
6000  HOME
6010  PRINT "  ADD ENTRIES"
6020  PRINT : PRINT "ENTER <$> TO QUIT, <RET> TO CONTI
     NUE:    ";Z$;: GET A$: PRINT A$
6030  IF A$ = "$" THEN 200
6040  IF N < 700 THEN 6080
6050  PRINT : PRINT B$ + "CANNOT ADD ENTRIES -- NO ROO
     M": PRINT : PRINT "THERE ARE ";NUM;" ENTRIES PLUS
      ";N - NUM;: PRINT " DELETED ENTRIES."
6060  PRINT : PRINT "BY SAVING THE FILE AND THEN RESTA
     RTING": PRINT "YOU CAN RECOVER THE SPACE HELD BY
     THE": PRINT "DELETED ENTRIES."
6070  PRINT : PRINT "PRESS RETURN TO CONTINUE ";Z$;: GET
     A$: PRINT : GOTO 200
6080  PRINT : PRINT "ENTRY NUMBER ";NUM + 1
6090  PRINT "   VOL: ";: INPUT VOL$
6100  PRINT "   NAME: ";: INPUT NAME$
6110  PRINT "   TYPE: ";: INPUT T$
6120  PRINT "   LENGTH: ";: INPUT L$
6130 NUM = NUM + 1:N = N + 1:PX(NUM) = N
6140 P$(N) =  RIGHT$ ("00" + VOL$,3) +  LEFT$ (NAME$ +
     BL$,30) +  LEFT$ (T$ + " ",2) +  RIGHT$ ("  " + L
     $,3)
6150  PRINT : PRINT "ANOTHER ENTRY? <Y> <N>: ";Z$;: GET
     A$: PRINT A$
6160  IF A$ = "Y" THEN 6040
6170  GOTO 200
7000  HOME
7010  PRINT "  DELETE ENTRIES"
7020  PRINT : PRINT "  <1> SELECT BY ENTRY NUMBER"
7030  PRINT "  <2> SELECT BY NAME"
7040  PRINT "  <3> QUIT"
7050  PRINT : PRINT "     CHOICE? ";Z$;: GET A$: PRINT
     A$:A =  ASC (A$) - 48
7060  IF A < 1 OR A > 3 THEN  PRINT B$;: GOTO 7050
7070  ON A GOTO 7080,7160,200
7080  PRINT : INPUT "ENTRY NUMBER? ";I
7090  IF I < 1 OR I > NUM THEN  PRINT B$;"DOES NOT EXI
     ST": GOTO 7080
7100  PRINT : PRINT "DELETE THIS ONE?"
7110  GOSUB 5140: PRINT Z$;: GET A$: PRINT A$: IF A$ <
      > "Y" THEN 7010
7120 NUM = NUM - 1: FOR K = I TO NUM:PX(K) = PX(K + 1)
     : NEXT
7130  PRINT B$ + "       DELETED!"
7140  GOTO 7010
7150  GOTO 7090
7160  PRINT : INPUT "ENTRY NAME? ";NAME$:L =  LEN (NAM
     E$):FLAG = 0:C = 0
7170  IF L = 0 THEN  PRINT B$;: GOTO 7160
7180  FOR J = 1 TO NUM:I = J - C
7190  IF  MID$ (P$(PX(I)),4,L) <  > NAME$ THEN 7230
7200 FLAG = 1: PRINT : PRINT "DELETE THIS ONE? <Y> <N>
     ": GOSUB 5140: PRINT Z$;: GET A$: PRINT A$
7205  IF A$ =  CHR$ (27) THEN 7010
7210  IF A$ <  > "Y" THEN 7230
7220 NUM = NUM - 1: FOR K = I TO NUM:PX(K) = PX(K + 1)
     : NEXT : PRINT B$;"          DELETED           !"
     :C = C + 1
7230  NEXT : IF FLAG = 0 THEN  PRINT : PRINT B$;"NOT F
     OUND"
7240  GOTO 7010
8000  HOME
8010  PRINT "  EDIT ENTRIES"
8020  PRINT : PRINT "MENU:"
8030  PRINT "  <1>  EDIT SINGLE ENTRY"
8040  PRINT "  <2>  EXAMINE ENTIRE FILE WITH OPTION": PRINT
           TO EDIT, DELETE OR SKIP"
8050  PRINT "  <3>  QUIT"
8060  PRINT : PRINT "     CHOICE? ";Z$;: GET A$: PRINT
     A$:A =  ASC (A$) - 48
8070  IF A < 1 OR A > 3 THEN  PRINT B$;: GOTO 8060
8080  ON A GOTO 8090,8190,200
8090  PRINT : PRINT "ENTER FILENAME TO BE EDITED ($ TO
      QUIT)": INPUT NAME$
8100  IF NAME$ = "$" THEN 8010
8110 L =  LEN (NAME$): IF L = 0 THEN  PRINT B$;: GOTO
     8090
8120 FLAG = 0: FOR I = 1 TO NUM
8130  IF  MID$ (P$(PX(I)),4,L) <  > NAME$ THEN 8170
8140 FLAG = 1: PRINT : PRINT "EDIT THIS ONE? <Y> <N>":
      GOSUB 5140: PRINT Z$;: GET A$: PRINT A$
```

continued on next page

SAVE TIME

when you change or debug an APPLESOFT® program!

PUT SOME SXR PLUS™ IN YOUR LIFE

- A Sorted Cross Reference utility that helps you modify, debug, optimize and document APPLESOFT programs.
- SXR PLUS tells you where variables, referenced line numbers, numeric constants and strings are used in your program.
- Tailor SXR PLUS to your needs while you access its many important features, including SEARCH.
- All information is in a single alphabetized list.
- Requires APPLE II,® 48K, DOS 3.3 (1 drive), APPLESOFT.
- Only $39.95 at your local computer store.

SXR PLUS is the newest product from

PRASEK COMPUTER SYSTEMS, INC.

P.O. Box 2427 Santa Clara CA 95051 ™
(408) 554-0420

DEALER INQUIRIES WELCOME

APPLE II and APPLESOFT are registered trademarks of APPLE COMPUTER INC

Apple Disk Librarian (Cont.)

```
8145  IF A$ =  CHR$ (27) THEN 8010
8150  IF A$ <  > "Y" THEN 8170
8160  GOSUB 8360
8170  NEXT : IF FLAG = 0 THEN  PRINT : PRINT B$;"NOT F
     OUND": GOTO 8090
8180  PRINT : PRINT B$;"THAT'S ALL": GOTO 8010
8190  HOME : PRINT "COMMANDS:  <E> EDIT,  <D> DELETE,"
     : PRINT "            <ESC> QUIT,  <RET> SKIP"
8200  PRINT "-----------------------------------------"

8210  POKE 34,3:C = 0
8220  FOR J = 1 TO NUM:I = J - C
8230  PRINT "ENTRY ";I;":"
8240  GOSUB 5140
8250  PRINT : PRINT "COMMAND? ";Z$;: GET A$: PRINT A$
8260  IF A$ =  CHR$ (13) THEN 8330
8270  IF A$ = "D" THEN 8320
8280  IF A$ =  CHR$ (27) THEN 8340
8290  IF A$ <  > "E" THEN  PRINT B$;: GOTO 8250
8300  GOSUB 8360
8310  GOTO 8330
8320 NUM = NUM - 1: FOR K = I TO NUM:P%(K) = P%(K + 1)
     : NEXT : PRINT : PRINT B$;"       DELETED!":C = C
      + 1: PRINT
8330  NEXT
8340  POKE 34,0
8350  GOTO 8010
8360  PRINT : PRINT "HERE IS THE COMPLETE ENTRY.  EDIT
      IT": PRINT "AS YOU WOULD A PROGRAM LINE USING ":
      PRINT "ESCAPE I,J,K, ETC. BE SURE NOT TO ": PRINT
     "CHANGE ITS LENGTH OR THE POSITION OF": PRINT "TH
     E FILETYPE OR LENGTH"
8370  PRINT : PRINT P$(P%(I)): PRINT : INPUT X$
8380 L1 =  LEN (X$): IF L1 < 38 THEN  PRINT : PRINT B$
     ;"TOO SHORT -- TRY AGAIN": GOTO 8370
8390  IF L1 > 38 THEN  PRINT : PRINT B$;"TOO LONG -- T
     RY AGAIN": GOTO 8370
8400 P$(P%(I)) = X$: PRINT
8410  RETURN
9000  HOME : PRINT "  EXIT"
9010  PRINT : PRINT B$;"BE SURE YOU HAVE SAVED YOUR CA
     TALOG.": PRINT : PRINT "DO YOU WANT TO EXIT NOW?
     <Y> <N>: ";Z$;: GET A$: PRINT A$: IF A$ <  > "Y" THEN
     200
9020  PRINT : PRINT B$"BYE-"B$"BYE"
9030  PRINT D$"MAXFILES 3": END
10000  GOTO 200: REM   RE-ENTRY VECTOR
```


MASTERCARD OR VISA!

Statistical Software!

For APPLE II+ or APPLE II w/ Language Card* Printer Optional

MBC Correlation Package

- Pearson, biserial, point biserial, phi, tetrachoric
- Spearman's Rho, Kendall's Tau, Goodman & Kruskal's Gamma
- Data Review & Editing
- Data Description Capability
- Statistical Evaluations of Correlations

$150.00

MBC Test Construction Pkg.

- Item Analysis
- Chronbach's Alpha
- Item Deletion & Addition
- Data Review & Editing

$70.00

Both Packages come with:

MBC File Mediator

- Permits use of existing sequential text files.

TO ORDER WRITE TO:

MOORE-BARNES COMPANY

P.O. Box 517 • Roseburg, OR 97470

or CALL TO ORDER: (503) 673-0345

* Apple II is a Trademark of Apple Computer, Inc.


```
SOURCE FILE: DLC
0000:              1 ************************************
0000:              2 *                                  *
0000:              3 *     DISK LIBRARIAN CATALOGER     *
0000:              4 *                                  *
0000:              5 *          COPYRIGHT 1983          *
0000:              6 *   BY MICROSPARC, INC             *
0000:              7 *                                  *
0000:              8 *   PROGRAM WRITTEN                *
0000:              9 *      BY :GEORGE D. PARKER        *
0000:             10 *                                  *
0000:             11 * THIS BINARY PROGRAM IS USED      *
0000:             12 * BY LIBRARIAN CATALOGER           *
0000:             13 * TO ACCESS THE CATALOG ON A       *
0000:             14 * STANDARD DOS 3.3 DISK IN ORDER   *
0000:             15 * TO READ IT INTO MEMORY           *
0000:             16 *                                  *
0000:             17 ************************************
0000:             18 *
------ NEXT OBJECT FILE NAME IS DLC.OBJ0
0300:             19          ORG  $300
0300:             20 **
0300:             21 **   ZERO PAGE VARAIBLES
0300:             22 **
00FD:             23 UERR     EQU  $FD        ; ERR FLAG FROM RWTS
00FE:             24 OFFSET   EQU  $FE        ; PTR TO FILE ENTRY
00FF:             25 FLAG     EQU  $FF        ; DETECT END OF CAT
0069:             26 VARTAB   EQU  $69        ; APPLESOFT VAR TABL PTR
0300:             27 **
0300:             28 **   IOB ADDRESSES
0300:             29 **
03D9:             30 RWTS     EQU  $3D9
B7EB:             31 VOL      EQU  $B7EB
B7EC:             32 TRACK    EQU  $B7EC
B7ED:             33 SECTOR   EQU  $B7ED
B7F0:             34 BUFR     EQU  $B7F0
B7F4:             35 CMD      EQU  $B7F4
B7FF:             36 ERR      EQU  $B7FF
0300:             37 **
0300:             38 **   RWTS SECTION
0300:             39 **
0300:             40 **   THIS IS THE FIRST ENTRY TO PROGRAM (CALL 768)
```

continued on next page

```
0300:                41 **
0300:A9 00           42 CALRW   LDA  #$00
0302:8D EB B7        43         STA  VOL
0305:8D F0 B7        44         STA  BUFR       ; BUFFER IS ABOVE HIMEM
0308:85 FD           45         STA  UERR       ; AND BELOW DOS BUFFER
030A:A9 99           46         LDA  #$99
030C:8D F1 B7        47         STA  BUFR+1
030F:A9 B7           48         LDA  #$B7       ; IOB TABLE
0311:A0 EB           49         LDY  #$EB
0313:20 D9 03        50         JSR  RWTS
0316:90 05           51         BCC  NEWTS
0318:AD FF B7        52         LDA  ERR        ; I/O ERROR IF CARRY SET
031B:85 FD           53         STA  UERR
031D:                54 **
031D:                55 **   THIS SECTION GETS TRACK,SECTOR
031D:                56 **   OF NEXT CATALOG SECTOR
031D:                57 **
031D:AD 01 99        58 NEWTS   LDA  $9901
0320:8D EC B7        59         STA  TRACK
0323:AD 02 99        60         LDA  $9902
0326:8D ED B7        61         STA  SECTOR
0329:60              62         RTS             ; END OF RWTS SECTION
032A:                63 **
032A:                64 **   THE REST OF THE PROGRAM READS
032A:                65 **   THE BUFFER AND LOADS APPLESOFT
032A:                66 **   VARIABLES
032A:                67 **
032A:                68 **   THIS IS THE SECOND ENTRY TO PROGRAM (CALL B10)
032A:                69 **
032A:A9 0B           70 LOOK    LDA  #$B        ; FIRST ENTRY STARTS
032C:85 FE           71         STA  OFFSET     ; IN BYTE $B
032E:20 00 03        72         JSR  CALRW
0331:A5 FD           73         LDA  UERR
0333:F0 01           74         BEQ  LOOK1
0335:60              75         RTS             ;  ERROR IN RWTS
0336:A6 FE           76 LOOK1   LDX  OFFSET
0338:BD 00 99        77         LDA  $9900,X
033B:85 FF           78         STA  FLAG
033D:D0 01           79         BNE  CHECK      ; IF FLAG = 0 WE ARE AT
033F:60              80         RTS             ; END OF CATALOG
0340:C9 FF           81 CHECK   CMP  #$FF
0342:B0 69           82         BCS  LOOPER     ; THIS ENTRY DELETED
0344:                83 **
0344:                84 **   THIS SECTION GETS LENGTH OF FILE
0344:                85 **
0344:A5 FE           86 BKIP    LDA  OFFSET
0346:18              87         CLC
0347:D8              88         CLD             ; BE SURE ITS BINARY ADD
0348:69 21           89         ADC  #$21
034A:AA              90         TAX
034B:BD 00 99        91         LDA  $9900,X    ; GET LENGTH LO/HI
034E:A0 0A           92         LDY  #$A
0350:91 69           93         STA  (VARTAB),Y ; STORE LENGTH HI/LO IN
0352:E8              94         INX             ; APPLESOFT VAR LNX
0353:88              95         DEY
0354:BD 00 99        96         LDA  $9900,X
0357:91 69           97         STA  (VARTAB),Y
0359:                98 **
0359:                99 **   THIS SECTION GETS THE FILE TYPE
0359:               100 **   AND LOADS IT AT THE END OF THE
0359:               101 **   FILE NAME
0359:               102 **
0359:               103 ** FIRST CHECK IF LOCKED
0359:               104 **
0359:A6 FE          105         LDX  OFFSET
035B:8A             106         TXA
035C:E8             107         INX
035D:E8             108         INX
035E:18             109         CLC
035F:D8             110         CLD
0360:69 22          111         ADC  #$22
0362:A8             112         TAY
0363:BD 00 99       113         LDA  $9900,X
0366:10 05          114         BPL  UNLOCK
0368:A9 2A          115         LDA  #$2A       ; LOAD A *
036A:4C 6F 03       116         JMP  PUT1
036D:A9 20          117 UNLOCK  LDA  #$20       ; LOAD A SPACE
036F:99 00 99       118 PUT1    STA  $9900,Y
0372:               119 **
0372:               120 **   NOW GET ACTUAL TYPE
0372:               121 **
0372:BD 00 99       122         LDA  $9900,X
0375:88             123         DEY
0376:A2 00          124         LDX  #$00
0378:29 7F          125         AND  #$7F       ; CLEAR BIT 7
037A:F0 04          126         BEQ  PUT2       ; IF ITS A TEXT FILE
037C:E8             127 LUP     INX             ; INCREMENT TABLE PTR
037D:4A             128         LSR  A
037E:90 FC          129         BCC  LUP        ; IF DON'T HAVE TYPE YET
0380:BD BD 03       130 PUT2    LDA  TABLE,X    ; LOAD TYPE
0383:99 00 99       131         STA  $9900,Y
0386:               132 **
0386:               133 **   THIS SECTION GETS FILE NAME,
0386:               134 **   SETS BIT-7 LOW, AND CONVERTS
0386:               135 **   CONTROL CHARACTERS TO NORMAL
0386:               136 **
0386:A0 1E          137         LDY  #$1E       ; FILENAME HAS 30 CHARS
```

continued on next page

we are not PIRATES

We simply prefer software that is NOT LOCKED UP!

If you do, too, then subscribe to hardcore computing

"HARDCORE Computing . . . warns pirates about the latest technology that companies are using against them." TIME, Feb. 8, 1982

"When some Apple enthusiasts . . . heard about the boycott (of bit-copy ads), they concluded that it was nothing but censorship and another example of the magazines ignoring the average Apple user to placate their advertisers. So they started their own publication, HARDCORE Computing . . ." ESQUIRE, Jan. 1982

Market your software.
Copy-protect your disks.
Normalize altered D.O.S.
Use bit-copiers to make back-ups.

PLUS Game, Utility, Business, Educational program listings.

For serious Apple computerists!

HARDCORE computing
Dept. N-1
P.O. Box 44549
Tacoma, WA 98444
(206) 581-6038

SUBSCRIPTION: $20 U.S.A. $29 Canada $33 Mexico $42 others
Sample copy: U.S.A. $5 elsewhere $8
U.S. Funds Only No Credit Cards

NAME _____
ADDRESS _____
CITY _____ ZIP _____
STATE _____

dBASE II®
BUSINE$$ APPLICATION SOFTWARE

•NOW IN OUR 3RD YEAR•
THE BUSINE$$
SOFTWARE SPECIALISTS
NOW OFFER YOU
AFFORDABLE dBASEII®
BUSINE$$ APPLICATION
SOFTWARE

•CASH ACCOUNTING• SYSTEM

Ideal for small Business or personal use where full double entry bookkeeping is not necessary. Extensively field tested in typical business situations.

FEATURES . . .

- Automatic check numbering with last check no., date, and current balance maintained in file
- Payee and Account Macros for rapid data entry of repetitive items
- Income and expenses can be allocated over multiple accounts
- Provision for cash transactions
- Monthly and Year-to-Date Income Reports with ratio analysis and net change in cash
- Produces monthly check register report
- Maintains checkbook balance
- Partial month income report available at any time during the current month
- Entirely menu driven
- Automatic set-up/install can be done without any programming knowledge
- Can be customized to suit individual applications
- Requires dBASE II (2.3 up)

INTRODUCTORY • SPECIAL $99 •

PACKAGE $PECIAL
• dBASE II + CASH ACTG. •
• $479 •
FORMATS: APPLE II - 8"SD
IBM PC-TELEVIDEO
MORE FORMATS COMING SOON-CALL!

COMING SOON
• G/L-A/R-A/P •
• INVOICING • INVENTORY •

• ORDER DESK •
(7:30 A.M. - 4 P.M. PAC. TIME M-F)
•CALL: (213) 823-0767•
•TECHNICAL HOTLINE: (213) 410-1986•
(7:30-9A.M. M-F-Consultation Available)

TERMS: All prices MAIL-ORDER ONLY - subject to change without notice.
PAYMENT: Cashiers CK/MO-Personal Checks allow up to 20 days-No COD or Terms-C.W.O. only. CA: Add 6% tax (LA: 6½%)
SHIPPING: Add: $3 UPS surface/$6 Blue Label (Postal/Foreign-Add $25 Postage)
• MASTERCARD / VISA •

* dBASEII-TM of Ashton-Tate PC02A/NIB01B

•FREDERICK E. DEEG•
and Associates
13234-A FIJI WAY
MARINA DEL REY, CA 90291

Apple Disk Librarian (Cont.)

```
0380:A6 FE      138          LDX  OFFSET
038A:E8         139          INX
038B:E8         140          INX
038C:E8         141 LUP2     INX
038D:BD 00 99   142          LDA  $9900,X
0390:29 7F      143          AND  #$7F
0392:C9 20      144          CMP  #$20       ; IF ASCII LESS THAN $20
0394:B0 02      145          BCS  OK         ; IT IS A CONTROL CHAR
0396:69 40      146          ADC  #$40       ; SO ADD $40 TO ASCII CODE
0398:9D 00 99   147 OK       STA  $9900,X
039B:88         148          DEY
039C:D0 EE      149          BNE  LUP2
039E:           150 **
039E:           151 **   THIS SECTION LOADS THE ADDRESS OF NAME$
039E:           152 **   WITH THE ADDRESS OF CORRECT PART
039E:           153 **   OF THE  INPUT BUFFER
039E:           154 **
039E:A0 03      155          LDY  #$3
03A0:A5 FE      156          LDA  OFFSET
03A2:18         157          CLC
03A3:69 03      158          ADC  #$3
03A5:91 69      159          STA  (VARTAB),Y
03A7:A9 99      160          LDA  #$99
03A9:C8         161          INY
03AA:91 69      162          STA  (VARTAB),Y
03AC:60         163          RTS             ; END OF ROUTINE
03AD:           164 **
03AD:           165 **   THIS IS THE THIRD ENTRY TO PROGRAM (CALL 941)
03AD:           166 **
03AD:18         167 LOOPER   CLC
03AE:D8         168          CLD
03AF:A5 FE      169          LDA  OFFSET
03B1:69 23      170          ADC  #$23       ; $23 BYTES PER CATALOG ENTRY
03B3:85 FE      171          STA  OFFSET
03B5:F0 03      172          BEQ  STEP       ; TO READ NEXT SECTOR
03B7:4C 36 03   173          JMP  LOOK1
03BA:4C 2A 03   174 STEP     JMP  LOOK
03BD:           175 **
03BD:           176 **   THIS SECTION HOLDS FILE TYPES
03BD:           177 **
03BD:D4 C9 C1   178 TABLE    ASC  "TIABSR"
03C0:C2 D3 D2

### SUCCESSFUL ASSEMBLY: NO ERRORS
```

```
SOURCE FILE: BUBBLE
0000:            1 ************************************
0000:            2 *                                  *
0000:            3 *            BUBBLE SORT           *
0000:            4 *        BY GEORGE D. PARKER       *
0000:            5 *                                  *
0000:            6 *          COPYRIGHT 1983          *
0000:            7 *        BY MICROSPARC, INC        *
0000:            8 *                                  *
0000:            9 * THIS BINARY PROGRAM PERFORMS     *
0000:           10 * THE SORTS REQUIRED BY THE        *
0000:           11 * LIBRARIAN FILER.                 *
0000:           12 *                                  *
0000:           13 ************************************
----- NEXT OBJECT FILE NAME IS BUBBLE.OBJ0
9900:           14          ORG  $9900      ; HIDDEN ABOVE HIMEM
9900:           15 **
9900:           16 **   APPLESOFT POINTERS
9900:           17 **
DFE3:           18 FNDVAR   EQU  $DFE3      ; LOCATES APPLESOFT VARIABLES
DEBE:           19 CHKCOM   EQU  $DEBE      ; SYNTAX CHECK FOR COMMA
9900:           20 **
9900:           21 **   ZERO PAGE POINTERS
9900:           22 **
0006:           23 USPTR    EQU  $06        ; ADDRESS OF POINTER TO UPPER STRING
0008:           24 LSPTR    EQU  $08
00F9:           25 UTGPTR   EQU  $F9        ; ADDRESS OF POINTER TO UPPER TAG (P%)
00FB:           26 LTGPTR   EQU  $FB
00EB:           27 USTR     EQU  $EB        ; ADDRESS OF UPPER STRING (P$)
00ED:           28 LSTR     EQU  $ED
9900:           29 **
9900:           30 **   PAGE THREE VARIABLES
9900:           31 **
03B0:           32 UTAG     EQU  $3B0       ; UPPER TAG (P%)
03B2:           33 LTAG     EQU  $3B2
03B4:           34 MAINCT   EQU  $3B4       ; MAIN LOOP COUNTER
03B6:           35 ARAYST   EQU  $3B6       ; START OF P$ POINTERS
03B8:           36 PTRZER   EQU  $3B8       ; P%(0)
03BA:           37 NUMEL    EQU  $3BA       ; DECIMAL 954,955
03BC:           38 PTRLST   EQU  $3BC       ; P%(N)
03BE:           39 CTR      EQU  $3BE
03C0:           40 TEMP     EQU  $3C0
03C1:           41 FLAG2    EQU  $3C1
03C2:           42 FLAG1    EQU  $3C2
9900:           43 **
9900:           44 **   NUMEL IS NUMBER OF ENTRIES TO SORT
9900:           45 **   FLAG INDICATES DOUBLE SORT
```

continued on next page

```
9900:                  46 **   OFFSET INDICATES FIRST CHAR TO COMPARE
9900:                  47 **   NCHRS INDICATES HOW MANY TO COMPARE
9900:                  48 **
9900:                  49 **   EACH IS PASSED BY APPLESOFT PROGRAM
9900:                  50 **
03C3:                  51 FLAG    EQU  $3C3       ; DECIMAL 963
03C4:                  52 OFFSET  EQU  $3C4       ; DECIMAL 964
03C5:                  53 NCHRS   EQU  $3C5       ; DECIMAL 965
03C6:                  54 NCHRS1  EQU  $3C6
9900:                  55 **
9900:                  56 **   LOCATE APPLESOFT ARRAYS
9900:                  57 **
9900:20 E3 DF          58         JSR  FNDVAR     ; A,Y HOLD ADDRESS OF LENGTH
9903:18                59         CLC             ; OF P$(0).  WE WANT ADDRESS
9904:69 01             60         ADC  #1         ; OF PTR FOR P$(0)
9906:8D B6 03          61         STA  ARAYST
9909:98                62         TYA
990A:69 00             63         ADC  #0
990C:8D B7 03          64         STA  ARAYST+1
990F:20 BE DE          65         JSR  CHKCOM     ; STRIP OFF COMMA
9912:20 E3 DF          66         JSR  FNDVAR     ; LOCATE ADDRESS OF P%(0)
9915:8D B8 03          67         STA  PTRZER
9918:8C B9 03          68         STY  PTRZER+1
991B:                  69 **
991B:                  70 **   INIT MAIN COUNTER
991B:                  71 **
991B:AD BA 03          72         LDA  NUMEL
991E:8D B4 03          73         STA  MAINCT
9921:AD BB 03          74         LDA  NUMEL+1
9924:8D B5 03          75         STA  MAINCT+1
9927:                  76 **
9927:                  77 **   FIND LAST P% ENTRY
9927:                  78 **
9927:0E B4 03          79         ASL  MAINCT     ; DOUBLE NUMBER OF
992A:2E B5 03          80         ROL  MAINCT+1   ; ENTRIES -- 2 BYTES FOR EACH
992D:18                81         CLC             ; P%(I)
992E:AD B8 03          82         LDA  PTRZER
9931:6D B4 03          83         ADC  MAINCT
9934:8D BC 03          84         STA  PTRLST
9937:AD B9 03          85         LDA  PTRZER+1
993A:6D B5 03          86         ADC  MAINCT+1
993D:8D BD 03          87         STA  PTRLST+1
9940:4E B5 03          88         LSR  MAINCT+1   ; RESET COUNTER TO
9943:6E B4 03          89         ROR  MAINCT     ; CORECT VALUE
9946:                  90 **
9946:                  91 ********************************
9946:                  92 *    START OF OUTER LOOP
9946:                  93 ********************************
9946:                  94 **
9946:38                95 BIGLP   SEC
9947:AD BC 03          96         LDA  PTRLST     ; ADDRESS OF P%(NUM)
994A:85 FB             97         STA  LTGPTR     ; INITIALLY ADDRESS OF P%(NUM)
994C:E9 02             98         SBC  #2
994E:85 F9             99         STA  UTGPTR     ; INITIALLY ADDRESS OF P%(NUM-1)
9950:AD BD 03         100         LDA  PTRLST+1
9953:85 FC            101         STA  LTGPTR+1
9955:E9 00            102         SBC  #0
9957:85 FA            103         STA  UTGPTR+1
9959:A9 00            104         LDA  #0
995B:8D C1 03         105         STA  FLAG2      ; INITIALIZE INTERCHANGE FLAG
995E:8D BE 03         106         STA  CTR        ; AND INNER LOOP COUNTER
9961:8D BF 03         107         STA  CTR+1
9964:                 108 **
9964:                 109 ********************************
9964:                 110 *    START OF INNER LOOP
9964:                 111 ********************************
9964:                 112 **
9964:A0 00            113 SMALLP  LDY  #0
9966:B1 FB            114         LDA  (LTGPTR),Y
9968:8D B3 03         115         STA  LTAG+1     ; ACTUAL VALUE OF LOWER TAG
996B:B1 F9            116         LDA  (UTGPTR),Y
996D:8D B1 03         117         STA  UTAG+1     ; ACTUAL VALUE OF UPPER TAG
9970:C8               118         INY
9971:B1 FB            119         LDA  (LTGPTR),Y
9973:8D B2 03         120         STA  LTAG
9976:B1 F9            121         LDA  (UTGPTR),Y
9978:8D B0 03         122         STA  UTAG
997B:                 123 **
997B:                 124 **   ARAYST HOLDS ADDRESS OF LENGTH
997B:                 125 **   AND PTR FOR P$(0).  TO GET SAME
997B:                 126 **   FOR P$(I) WE MUST ADD 3*I TO THE
997B:                 127 **   ADDRESS IN ARAYST.  I WILL BE
997B:                 128 **   EITHER UPPER OR LOWER TAG.
997B:                 129 **
997B:A2 02            130         LDX  #2
997D:A0 03            131 LOOP2   LDY  #3
997F:AD B6 03         132         LDA  ARAYBT
9982:95 06            133         STA  USPTR,X    ; X=2 GIVES LSPTR
9984:AD B7 03         134         LDA  ARAYST+1
9987:95 07            135         STA  USPTR+1,X
9989:18               136 LOOP3   CLC
998A:B5 06            137         LDA  USPTR,X
998C:7D B0 03         138         ADC  UTAG,X
998F:95 06            139         STA  USPTR,X
9991:B5 07            140         LDA  USPTR+1,X
9993:7D B1 03         141         ADC  UTAG+1,X
9996:95 07            142         STA  USPTR+1,X
```

continued on next page

DI$COUNT
APPLE® BUSINE$$
•SOFTWARE•HARDWARE•
•NOW IN OUR 3RD YEAR•

•$AVE TO 65%•800+ PGMS-CALL•

—ACCOUNTING SOFTWARE—

	LIST	COST
•Peachtree Series 40•		
G/L-A/R-A/P-INV-Payroll (Ea)	$400	$229
•B.P.I. G/L-A/R-Pay-Time Actg (Ea)	395	279
•Systems + /S.W. Dim. Accounting +][		
G/L-A/R-A/P-Inventory (Ea)	395	235
ALL Four Modules	1250	849
•Continental Software		
G/L-A/R-A/P-Payroll (Ea)	250	154
ALL Four Modules	1000	605
Home Accountant	75	47

—COMMUNICATIONS SOFTWARE—

•SW Data Z-Term Professional•	150	95
ASCII Express Professional	130	85
•SE Software Data Capture 4.0	65	43

—DATA BASE/FINANCIAL MOD.—

•Microsoft MultiPlan (Spec. DOS or CPM)	275	169
•Stoneware DB Master	229	142
Utility Paks (Ea)	99	59
•Visicorp Visicalc (Ea)	250	165
•Ashton-Tate dBASE II (Full ver.)•	700	399
•Fox/Geller dUTIL•	99	55
Quickcode•	295	179
dGRAPH•	295	179
•DATABASE PACKAGE $PECIAL•		
dBASE II (Full ver.) + Quickcode	995	559
•Sorcim Supercalc•	295	179
•Software Publishing PFS (New)	125	79

—SPECIALIZED SOFTWARE—

•HowardSoft Tax Preparer 1983	225	139
•Applied Software Versa Form	389	245

—UTILITY SOFTWARE—

•Nibbles Away][(Copy pgm)	70	55
•Sensible Super Disk Copy III	30	22
•Microsoft TASC Compiler	175	115

—WORDPROCESSING SOFTWARE—

•Perfect S/W Perfect Writer•	495	189
•Sierra On-Line Screenwriter II	130	79
•Kensington Format II	250	159
•Artsci Magic Window II (40/80)	150	95
•SofSys Executive Secretary	250	165
•Muse Supertext 40/56/70	125	95
•LJK Letterperfect w/mail-merge	150	95
•Silicon Valley Word Handler	199	129

—HARDWARE $PECIALS—

•Microsoft 16K RAMCARD	195	89
Premium System	695	449
•Videx Videoterm w/Soft-Switch	380	239
Keyboard Enhancer][	149	99
•Kensington SYSTEM SAVER Fan	90	63
•Novation 212 Apple Cat	725	579
•D.C. Hayes Micromodem][	379	255
Smartmodem 1200	699	499
•MICRO-SCI A2 35T DISK DRIVE	395	289
•N.E.C. 12" Green Hi-Res CRT	285	169
•Practical Peripherals MB][(32K)	295	209
•Microtek Buffered Dumpling (32K)	209	195
•Corona 5mB Hard Disk		$Call

—CP/M™ HARDWARE PACKAGES—

•ALS 6mHz/64K CP/M (V3.0) Cd & Videx 80col/s.w.•
•PACKAGE $PECIAL $529 • $AVE $250•
•Per. Computer Prod. APPLI-CARD (6 mHz)•
Features: Z-80 w/CP/M•64K•70 col. video•
•$PECIAL $409-$AVE $186•

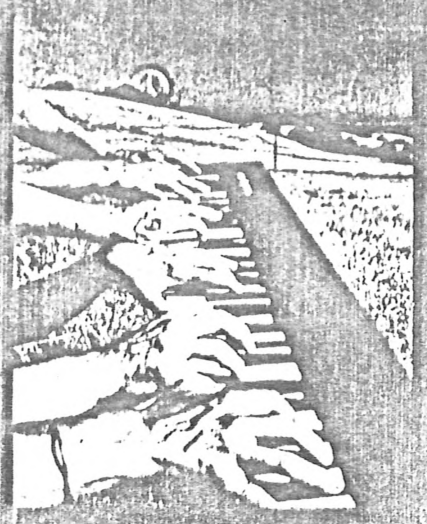
•ORDER DESK•
TO ORDER (7:30AM-4PM M-F)—CALL...
•(213) 823-0767•
WE PAY FOR YOUR CALL!
PHONE ORDERS OVER $300 DEDUCT $3
•TECHNICAL HOTLINE: (213) 410-1986•
•7:30-9AM M-F•Consultation available•

TERMS: MAIL-ORDER ONLY. Prices subject to change w/o notice - Some items-limited supply. Mail Orders under $100—Add $10.
PAYMENT: Cashier's Ck/MO/Bank Transfers/Personal cks-Allow up to 20 days-CA: Add 6% Tax. (LA: 6½%) - NO COD or Terms - C.W.O. only.
SHIPPING: UNDER $750-Add 2% (min. $5) UPS Surface
Over $750-FREE UPS SURFACE!
Postal/Foreign Add $25 + Postage
MASTERCARD/VISA: Add 2% (Min. $200 charge)

•CP/M-TM Digital Research •Apple-TM Apple
•Pgm requires CP/M• NIB01/POC03

•FREDERICK E. DEEG•
and Associates
13234-A FIJI WAY
MARINA DEL REY, CA 90291

live!
alphaSyntauri™



Live music on your Apple* computer?

Yes! the alphaSyntauri™ a hands-on digital musical instrument, the first truly soft instrument, puts you in command...

4 or 5-octave keyboard with software for:

- **Creating**—instruments, sound effects
- **Performing**—8 voices, split keyboard, sustain and portamento footpedals
- **Recording**—16 independent tracks, sequencing, speed control
- **Learning**—music theory and keyboard skills
- **Composing**—polyphonic score print-out of recordings*

*available December 1982

Tell me more!

name _____

address _____

_____

state zip

phone (___) _____

the Sounds of Science, from Syntauri Ltd., Dept. NM1
3506 Waverley Street
Palo Alto, CA 94306

Call your local Apple* dealer about a hands-on demonstration.

*trademark of Apple Computer, Inc.

Apple Disk Librarian (Cont.)

```
9990:88          143          DEY
9991:D0 EE       144          BNE   LOOP3
9993:CA          145          DEX
9994:CA          146          DEX
9995:F0 DE       147          BEQ   LOOP2      ; FIRST LSPTR, THEN USPTR
9997:            148 **
9997:            149 **    NOW WE GET THE ACTUAL ADDRESS OF
9997:            150 **    THE STRINGS TO BE COMPARED
9997:            151 **
9997:A0 01       152          LDY   #1
9999:B1 06       153 LOAD     LDA   (USPTR),Y
999B:99 EB 00    154          STA   USTR,Y
999E:B1 08       155          LDA   (LSPTR),Y ; GET ADDRESS OF LOWER STRING
99A0:99 ED 00    156          STA   LSTR,Y
99A3:88          157          DEY
99A4:10 F3       158          BPL   LOAD
99A6:            159 **
99A6:            160 **    TIME TO COMPARE CHARACTERS
99A6:            161 **
99A6:AC C4 03    162 CMPSTR   LDY   OFFSET     ; START COMPARISON AT THIS BYTE
99A9:AD C5 03    163          LDA   NCHRS      ; COMPARE THIS MANY
99AC:18          164          CLC
99AD:6D C4 03    165          ADC   OFFSET
99B0:8D C6 03    166          STA   NCHRS1
99B3:AD C3 03    167          LDA   FLAG       ; INDICATES IF DOUBLE SORT
99B6:8D C2 03    168          STA   FLAG1
99B9:B1 ED       169 CHREQ    LDA   (LSTR),Y  ; COMPARE CHARACTERS
99BB:D1 EB       170          CMP   (USTR),Y
99BD:D0 17       171          BNE   NOTEQL
99BF:C8          172          INY
99C0:CC C6 03    173          CPY   NCHRS1     ; IF ALL EQUAL CHECK FLAG
99C3:D0 F4       174          BNE   CHREQ
99C5:A9 03       175          LDA   #3         ; TO SEE IF VOLUME IS TO BE CHECKED
99C7:CD C2 03    176          CMP   FLAG1
99CA:D0 24       177          BNE   NOFLIP
99CC:8D C6 03    178          STA   NCHRS1
99CF:A0 00       179          LDY   #0
99D1:8C C2 03    180          STY   FLAG1
99D4:F0 E3       181          BEQ   CHREQ
```

continued on next page

BUSINESS PEOPLE · PROFESSIONALS · HOBBYISTS
All Users of the Apple II* Computer

LETS GET ORGANIZED!!

WITH THE PERFECT COMPLEMENT TO YOUR SOFTWARE

SPECIAL FEATURES:

- AUTOMATIC LABELLING WITH JUST A FEW KEYSTROKES
- TYPING NOT NECESSARY
- MENU DRIVEN
- 4" OR 5" LABEL CAN BE USED

PRICE $59.95

PACKAGE INCLUDES
- LABELLER FOR DOS 3.3
- 300 5" LABELS
  (LABELLER FOR DOS 3.2 AVAILABLE ON SPECIAL ORDER)

A "MUST" FOR
CORPORATIONS · ACCOUNTANTS
LAWYERS · DOCTORS
DENTISTS · SCHOOLS
ENGINEERS
IN FACT, FOR EVERYONE

REQUIRES 48K APPLE II* OR APPLE II+* LANGUAGE OR SIMILAR CARD · DISK DRIVE · PRINTER

SEND CHECK OR MONEY ORDER TO:

PRACTICAL SOFTWARE LTD.
P.O. Box 3000 Dept. N Pomona, N.Y. 10970
Phone: 914-425-1158

PLEASE ADD $3.00 FOR SHIPPING AND HANDLING
N.Y. STATE RESIDENTS: ADD APPLICABLE SALES TAX

*APPLE II AND APPLE II+ ARE REGISTERED TRADEMARKS OF THE APPLE COMPUTER, INC.

```
99DE:D0 1A      182 NOTEQL  BCB   NOFLIP
99E0:A0 01      183         LDY   #1
99E2:           184 **
99E2:           185 **    IF THE LOWER STRING SHOULD COME
99E2:           186 **    FIRST, WE MUST BUBBLE IT UP
99E2:           187 **
99E2:B1 FB      188 FLIP    LDA   (LTGPTR),Y
99E4:8D C0 03   189         STA   TEMP
99E7:B1 F9      190         LDA   (UTGPTR),Y
99E9:91 FB      191         STA   (LTGPTR),Y
99EB:AD C0 03   192         LDA   TEMP
99EE:91 F9      193         STA   (UTGPTR),Y
99F0:88         194         DEY
99F1:10 EF      195         BPL   FLIP
99F3:A9 01      196         LDA   #1          ; SET FLAG TO INDICATE
99F5:8D C1 03   197         STA   FLAG2       ; A BUBBLE OCCURRED
99F8:           198 **
99F8:           199 **    IF UPPER STRING FIRST WE DO NOT
99F8:           200 **    PERFORM A BUBBLE
99F8:           201 **
99F8:EE BE 03   202 NOFLIP  INC   CTR         ; INCREMENT INNER LOOP COUNTER
99FB:D0 03      203         BNE   NXTPR
99FD:EE BF 03   204         INC   CTR+1
9A00:AD BF 03   205 NXTPR   LDA   CTR+1
9A03:CD B5 03   206         CMP   MAINCT+1    ; ARE WE DONE WITH INNER LOOP?
9A06:90 0E      207         BCC   BUBBLE
9A08:F0 02      208         BEQ   CHK2
9A0A:B0 1E      209         BCS   NOBUB
9A0C:AD BE 03   210 CHK2    LDA   CTR
9A0F:CD B4 03   211         CMP   MAINCT
9A12:90 02      212         BCC   BUBBLE
9A14:B0 14      213         BCS   NOBUB
9A16:38         214 BUBBLE  SEC               ; NOT DONE SO UPPER TAG PTR
9A17:A5 F9      215         LDA   UTGPTR      ; BECOMES LOWER TAG PTR AND
9A19:85 FB      216         STA   LTGPTR      ; WE GET NEW UPPER TAG PTR
9A1B:E9 02      217         SBC   #2
9A1D:85 F9      218         STA   UTGPTR
9A1F:A5 FA      219         LDA   UTGPTR+1
9A21:85 FC      220         STA   LTGPTR+1
9A23:E9 00      221         SBC   #0
9A25:85 FA      222         STA   UTGPTR+1
9A27:4C 64 99   223         JMP   SMALLP      ; GO THROUGH INNER LOOP AGAIN
9A2A:           224 **
```

continued on next page

IF YOUR COMPUTER'S IMPORTANT TO YOU

Protect It!

Without SAFEWARE,™ you could be uninsured. For as little as $35 a year, SAFEWARE provides complete protection for all hardware, media and purchased software. Both business and home application. Call toll free today for more information or immediate protection. Columbia National General Agency, 88 E. Broad, Columbus, Ohio 43215. *(In Ohio call 1-800-848-2112)*

1-800-848-0598

EARLY GAMES

FOR YOUNG CHILDREN

Nine educational and entertaining games controlled by a single program. Even very young children can select a game, play it, and select a different game...ALL BY THEMSELVES!

- PICTURE MENU GIVES CHILDREN CONTROL
- MATCH NUMBERS AND LETTERS
- COUNT COLORFUL BLOCKS
- ADD AND SUBTRACT STACKS OF BLOCKS
- LEARN THE ALPHABET
- PRACTICE SPELLING NAMES
- COMPARE SHAPES
- DRAW AND SAVE COLORFUL PICTURES

The large numbers and letters fill the screen with color. Children enter single key stroke responses and get immediate visual and musical feedback. Hints are provided when appropriate. Beyond just teaching children basic skills, EARLY GAMES makes them feel comfortable as they control the computer. Designed for children ages 2½ to 6 years old.

EARLY GAMES offers the child a diverse selection of activities which stimulate the process of problem solving as well as foster individual creativity.

Pamela Bach, Director
Youth World Day Care Center

I took EARLY GAMES home for my kids and they really liked it! It held their attention and they learned from it!

Jeanette Fritze
Computer Saleswoman

EARLY GAMES can help children learn new concepts, information, and skills and also introduce them to the joys and benefits of home computers.

Peter Clark, faculty
Institute of Child Development
University of Minnesota

All nine games for $29.95
(Minnesota residents add 5% sales tax)

Apple II Plus
IBM Personal Computer
Atari 24K Disk or 16K Cassette
TRS-80 Model I/III 32K Disk or 16K Cassette
TRS-80 Color Computer 16K Disk or Cassette

VISA MasterCard

VISA/MasterCard

EARLY GAMES
educational software

Suite 140B
Shelard Plaza North
Minneapolis, MN 55426
1-800-328-1223

Minnesota residents call:
612-544-4720

Apple Disk Librarian (Cont.)

```
9A2A:                225 ********************************
9A2A:                226 *    END OF INNER LOOP
9A2A:                227 ********************************
9A2A:                228 *
9A2A:AD C1 03        229 NOBUB   LDA  FLAG2      ; INNER LOOP DONE SO
9A2D:C9 01           230         CMP  #1         ;CHECK TO SEE IF ANY FLIPS
9A2F:90 1E           231         BCC  DONE       ; IF NOT EXIT OUTER LOOP
9A31:38              232         SEC
9A32:AD B4 03        233         LDA  MAINCT     ; IF NOT DONE, DECREMENT
9A35:E9 01           234         SBC  #1         ; OUTER LOOP COUNTER.
9A37:8D B4 03        235         STA  MAINCT
9A3A:AD B5 03        236         LDA  MAINCT+1
9A3D:E9 00           237         SBC  #0
9A3F:8D B5 03        238         STA  MAINCT+1
9A42:AD B5 03        239         LDA  MAINCT+1
9A45:D0 05           240         BNE  CONT
9A47:AD B4 03        241         LDA  MAINCT
9A4A:F0 03           242         BEQ  DONE       ; IF MAIN COUNTER NOT 0
9A4C:4C 46 99        243 CONT    JMP  BIGLP      ; REPEAT OUTER LOOP
9A4F:                244 *
9A4F:                245 ********************************
9A4F:                246 *    END OF OUTER LOOP
9A4F:                247 ********************************
9A4F:                248 *
9A4F:A9 B7           249 DONE    LDA  #$B7       ; RING THE BELL, WE'RE DONE!
9A51:20 ED FD        250         JSR  $FDED
9A54:60              251         RTS

### SUCCESSFUL ASSEMBLY: NO ERRORS
```


DOUBLE your DISKETTES

The only reasons your Apple][cannot use the back side of your diskette are:

1. There is no notch.
2. The diskette manufacturer did not test the back side, or worse, put the flawed front to the back.

A nibbling tool will solve problem number 1.

DISK PREP will solve problem number 2.

DISK PREP formats and tests your disk. Sectors with flaws are left so that they cannot be used. Your disk is left ready to boot, complete with a flaw report program saved on it.

$25.00

sympathetic software
9531 Telhan Drive
Huntington Beach, CA 92646

California residents add $1.50 sales tax

Dealer inquiries invited

Repair Blown Disks in Minutes

Remember the sinking feeling you got the last time you wiped out a disk? With **DARK**, you'll never have that feeling again because **DARK** repairs damaged disks quickly and easily. What you can do with **DARK** in just a few minutes would take an experienced programmer hours to do!

- user friendly - no technical knowledge needed
- fixes files which can't be read... no more I/O ERRORs!
- completely rebuilds a disk, even if the catalog can't be read.
- displays files in several meaningful formats: BASIC program, text file, hi-res picture, etc.
- includes handy utilities to format a track, search disks, map disks, etc
- requires 48K Apple II or Apple II Plus and DOS 3.3

DARK The Disk Repair Kit only $34.95

Load Programs in 1/5th the Time

Cut the time it takes to load your disk files by as much as 80% with **FASTLOAD**. Load a hi-res picture in just three seconds; load a 32K program in six! **FASTLOAD** is an update to DOS 3.3 which speeds up the LOAD, RUN, BLOAD and BRUN commands. The updated DOS cannot INITialize disks, but is otherwise completely compatible with DOS 3.3 files and programs.

- available on an unprotected disk
- compatible with all programs which don't need the INIT command
- can be used without modifying or reformatting existing disks
- disk includes the following programs: **FASTLOAD, FASTOFF** and **UPDATE DOS**
- requires an Apple II or Apple II Plus and DOS 3.3

FASTLOAD from microseeds only $24.95

microseeds
LAKEVIEW TERRACE, STAFFORD SPRINGS, CT 06076
(203) 872-0917

'Apple II' is a trademark of Apple Computer, Inc.
'FASTLOAD' and 'DARK' are trademarks of microseeds