$1

Washington Apple Pi

Volume 2 August 1980 Number 8

Highlights

In This Issue

Officers & Staff

President	-Bernard Urban	(301) 229-3458
Vice President	-Rich Wasserstrom	(703) 893-9147
Treasurer	-Bob Peck	(301) 468-2305
Secretary	-Dana Schwartz	(301) 725-6281
Members-at-Large	-Hersh Pilloff	(301) 292 3100
	-Mark Crosby	(202) 488-1980
	-Sandy Greenfarb	(301) 674-5982
Editor	-Bernard Urban	(above)
Associate Editor	-Rich Wasserstrom	(above)
	-Genevie Urban	(301) 229-3458
Librarian	-David Morganstein	(301) 972-4263

Washington Apple Pi
P. O. Box 34511
Washington, D.C. 20034
(301) 468-2305

Membership dues for Washington Apple Pi are $12.00 per calendar year. If you are interested in joining our club, call our number and leave your name and address. An application form will be mailed to you. Or if you prefer, write us at the above PO Box.

EVENT QUE

Washington Apple Pi meets on the 4th Saturday of each month at 9:30 AM at George Washington University School of Engineering, 23rd and H Streets, N.W. Call the club telephone for exact meeting location.

NOVAPPLE meets on the 2nd Wednesday at 7:30PM at Computers Plus in Franconia, and on the 4th Thursday at 7:30 PM at Computerland of Tysons Corner.

CLASSIFIEDS

For Sale: 32K Memory, $95, or 16K, $50. Wanted: used floppy disk drive and controller. Ira Cotton, (H) 468-2266, (O) 951-2200.

GROUP PURCHASE POWER--DOS 3.3

---A membership benefit---

Those members who now have a Disk II will certainly want to upgrade their systems with DOS 3.3. Normally selling for $60 plus tax, your club will offer it for a substantial discount. Your deposit check for $20 is required before an order can be placed. For further information, call the club number, (301) 468-2305.

Classified ads accepted from members 50 words or less at no charge provided the material is obviously non-commercial. Submit your classified at least 30 days in advance attention CLASSIFIED ADS, PO Box 34511, Washington, DC 20034.

MINUTES

The Washinton Apple Pi meeting of July 26, 1980 was called to order at 9:35 a.m. by the Vice President. It was announced that a new meeting format would be inaugurated at this meeting, which would consist of a short business meeting followed by several concurrent group meetings. It is hoped that this new structure will better serve the wide-ranging needs of our membership. Any comments or suggestions would be greatly appreciated by the officers.

Dave Morganstein reminded the members that original contributors to the library would be given a free disk of their choice. The President announced that the club would have a table at a computer fair being held in New York City by the Big Apple computer club on August 16 and that all members are invited to attend. Dave Efron renewed his request for reviewers of Apple related articles.

The meeting was then adjourned into smaller sections.

Dana J. Schwartz, Secretary

NEWSIG NOTES
by Sara LaVilla

The New Users' Group (NEWSIG) held its first meeting following Apple Pi's July 26 general meeting. Al Weiner has agreed to lead this group of Apple newcomers through the basics.

This first meeting consisted of an exchange of questions and answers. NEWSIG extends its thanks to David Morganstein and John Moon who lent their expertise to the group and supplied answers to our questions.

Since these questions and answers may be of interest to new users who were unable to attend the meeting, they are reproduced below:

Q. What are SIGS?
A. Subgroups of Washington Apple Pi which have special interests (games, machine language, Pascal, new users).

Q. What does the POP command do?
A. When you have several nested subroutines the Apple stores the return addresses in a stack of these addresses, the latest being on top. When a return command is encountered, the topmost address in the stack is removed and execution proceeds from that line number. In Integer Basic POP may substitute for RETURN in a subroutine. POP, however, wipes out the top line number being stored, picks up the next and removes it with execution proceeding fron that line number.

Q. Is it advisable to order from discount houses through the mail?
A. As in most of life's endeavors, you get what you pay for. If you cannot perform your own service, or if you will need to rely on a local dealer's assistance with hard or software, then you probably should purchase locally from a "full service" dealer. If, on the other hand, you are confident of your repair and software abilities, or are willing to pay for such services in the event of disaster, then you may wish to try one of the more reputable mail order houses. Ask several experienced club members to recommend mail order houses. Never buy software sight unseen unless recommended by a friend or reputable computer publication.

Q. What is IAC?
A. International Apple Core

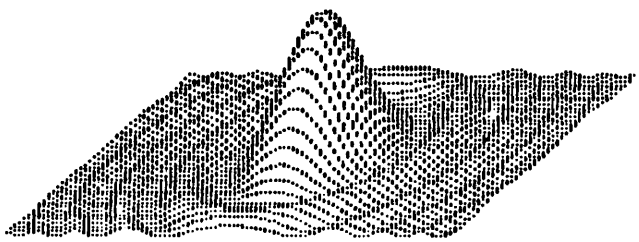
Q. What are the advantages of DOS 3.3?
A. DOS 3.3 permits approximately 20% more data storage per diskette then than 3.2.1. A disadvantage is that current canned disks won't work on 3.3 and there is some question about PASCAL compatibility.

Q. How do you perform TEXT and HOME in Integer Basic?
A. CALL-1123 for TEXT, and CALL-936 for HOME.

NEWSIG will try to develop a glossary of terms for new users. NEWSIG asked that Apple Pi officers and other members speaking at meetings identify themselves.

Recommended Purchases:

Apple Monitor Peeled @$15.00
CRAE @$20.00
Call A.P.P.L.E
Newsletter



ASMSIG - Assembly Language Special Interest Group

The ASMSIG met for the first time following Apple-Pi's regular July meeting. Twelve curious members showed up - each with his own purposes, objectives, and attitudes. Backgrounds varied from some who have had decades of assembly programming experience (clearly not all on the APPLE), to those of us who are just beginning to learn the difference between JMP and JSR. And interests were as widely ranged: from a hook into DOS (and other ROM code), to computer typesetting, high speed communications, higher precision floating point arithmatic and trigonometry, and even a full blown image processing system!!

It would seem that regardless of our backgrounds, irrespective of our specific interests, we each have similar concerns - and I offer my own interpretation of those concerns:

:First of all - If things can be done in a higher level language - do it!

:But all things cannot be done in an interpreted BASIC or PASCAL, - especially when time is a crucial factor. Thus there are occasions when ASM is a necessity.

:Necessity or not, Assembly language should be seen as just another tool in the programmer's kit bag. A tool, neither mysterious nor arcane, to be used when applicable.

:Philosophy aside, we are committed to spend as little time as possible reinventing the 'wheel'. Our purpose as a group is to share insights, build macros, and establish a sharable library of utilities.

:Standards- We have a limited agreement to use TED-II+ (or any other version of TED) found on the APPLE-PI disk #8. That way we can most easily share our stupendous and astonishing results!

:As we struggle with TED, we are agreed on a search for a more capable assembler - one which can handle macros and relocatable code.

:More urgently we must soon establish APPLE-PI standards concerning
-initializtion
-page usage (especially page zero!)
-documentation
-interrupt protection
-stack usage
-????

After an extremely informal survey of the group, here's an informal expectation of potential articles which could appear in future newsletters:
:Peeking at DOS - A Continuing Mystery
:6502 ASCII to Baudot
:FASTRES
:More TED Documentation
:Floating Point Routines
:Fast Fourier Transorms - Simple as Apple-Pi
:PASCAL Probes & Latches

We will meet again after the next regular meeting. Newcomers welcome!!

SAVE TAPE

by Dana J. Schwartz

Have you ever wanted to back-up your irreplaceable disks, but didn't have enough extra blank disks to hold them all? Was saving each program individually on tape too time consuming? What if you could save an entire disk in less than ten minutes on tape? Interested?

The following three routines will save all the occupied sectors of any disk on your cassette tape, using all available RAM for buffers. Since the sectors are saved in several large groups of buffers (maximum of 132 at a time on a 48K machine), the entire process will take ten minutes or less, depending on how full the disk is. The data may be restored at a later time using a routine which is also placed on tape.

The user creates an EXEC file using "MAKE SAVE TAPE EXEC'. This file places the restore routine "RESTP' at the start of the tape, and then starts the save process (routine "SVTP'). The user must have the tape ready to go when "SAVE TAPE' is EXEC'd and will be prompted when to insert the disk to be saved. At several points the user will be told to stop (or pause) the tape while sectors are read from the disk.

To restore the data, a blank initialized disk is booted, HIMEM is set to 4096, and the first file (the restore routine) is read from tape using the LOAD command. The tape is then stopped (or paused) and this routine is executed (using RUN). The user will again be prompted when to start and stop the tape as the disk is restored. Note that this process must be done on a system with at least as much RAM as the system on which the tape was made, since the buffer size on tape is fixed.

```
]LIST
  10 D$="": REM CTRL-D
  20 PRINT D$;"OPEN SAVE TAPE"
  30 PRINT D$;"WRITE SAVE TAPE"
  40 PRINT "LOAD RESTP"
  50 PRINT "MON I"
  60 PRINT "REM START TAPE IMMEDIATEL
     Y"
  70 PRINT "SAVE"
  80 PRINT "REM STOP TAPE"
  90 PRINT "NOMON I"
 100 PRINT "HIMEM:4096"
 120 PRINT "RUN SVTP"
 130 PRINT D$;"CLOSE SAVE TAPE"
 140 END
1000 REM ************************
1010 REM * MAKE SAVE TAPE EXEC *
1020 REM * BY DANA J. SCHWARTZ *
1030 REM *       6/1/80        *
1040 REM ************************

]
```

```
]LIST
  10 DIM C$(5)
  20 IF PEEK (76)=0 AND PEEK (77
     )=16 THEN 40
  30 PRINT "TYPE 'EXEC SAVE TAPE'"
     : END
  40 DIM HEX$(75): GOTO 60
  50 HEX$( LEN( HEX$ )+1 )=" N E88AC"
     : FOR I=1 TO LEN( HEX$ ): POKE
     511+I, ASC( HEX$(I) ): NEXT I:
     POKE 72,0: RETURN
  60 HEX$="1000: A9 10 A0 12 20 D9 03
     B0 01 60 AD 1F 10 8D 11 10"
     : GOSUB 50: CALL -144
  70 HEX$="1010: 60 00 01 60 01 00 11
     00 24 10 00 09 00 00 01 00"
     : GOSUB 50: CALL -144
  80 HEX$="1020: 00 60 01 00 00 01 EF
     D8": GOSUB 50: CALL -144
  90 TOP= PEEK (978)-7
 100 INPUT "INSERT DISK TO BE SAVED A
     ND HIT RETURN",C$
 110 TR=17:SE=0:BUF=17: GOSUB 2000
     : REM GET VTOC
 120 POKE 4224,TOP-18: REM MAX FOR
     RESTP
 130 PRINT "SAVING VTOC": INPUT
     "START TAPE AND HIT RETURN"
     ,C$
 140 POKE 60,0: POKE 61,16: POKE
     62,255: POKE 63,17: CALL -307
     : REM SAVE RWTS/VTOC
 150 PRINT "STOP TAPE"
 160 BUF=18
 170 FOR TR=3 TO 34
 180 B1= PEEK (4352+56+4*TR):B2=
      PEEK (4352+57+4*TR)
 190 BM=B1*32+B2/8
 200 FOR SE=12 TO 0 STEP -1
 210 IF BM>4095 THEN 260: REM NOT
     USED
 220 GOSUB 2000: REM READ SECTOR
 230 BUF=BUF+1: IF BUF<TOP THEN
     260
 240 GOSUB 1000: REM WRITE TAPE
 250 BUF=18
 260 BM=( BM*2 ) MOD 8192
 270 NEXT SE
 280 NEXT TR
 290 IF BUF>18 THEN GOSUB 1000: REM
     PUT OUT LAST GROUP
 300 PRINT "SAVE TAPE COMPLETE"
 310 END
1000 PRINT "WRITING ";BUF-18;" BUFFER
     S"
1010 INPUT "START TAPE AND HIT RETURN
     ",C$
1020 POKE 60,0: POKE 61,18: POKE
     62,255: POKE 63,BUF-1: CALL
     -307
1030 PRINT "STOP TAPE"
1040 RETURN
2000 POKE 4118,TR: POKE 4119,SE:
      POKE 4123,BUF: POKE 4113,0

2010 PRINT "TR=";TR;" SE=";SE
2020 CALL 4096
2030 IF PEEK (4113)=0 THEN RETURN

2040 PRINT "ERROR", PEEK (4113)
2050 IF TR=17 AND SE=0 THEN END
     : REM ERROR ON VTOC
2060 RETURN
10000 REM ************************
10010 REM *  SAVE TAPE CREATE   *
10020 REM * BY DANA J SCHWARTZ *
10030 REM *       6/1/80        *
10040 REM ************************

]
```

```
]LIST
  10 DIM C$(5),NR(3)
  20 IF PEEK (76)=0 AND PEEK (77
     )=16 THEN 40
  30 PRINT "HIMEM:4096 AND RE-RUN"
     : END
  40 PRINT "READING VTOC": INPUT
     "START TAPE AND HIT RETURN"
     ,C$
  50 POKE 60,0: POKE 61,16: POKE
     62,255: POKE 63,17: CALL -259
     : REM READ RWTS/VTOC
  60 PRINT "STOP TAPE"
  70 PRINT "RESTORING VOLUME "; PEEK
     (4352+6)
  80 MAX= PEEK (4224):TOP=MAX+18

  90 GOSUB 3000:RD=0: REM GET INPUT
     SIZES
 100 POKE 4126,2
 110 PRINT "INSERT NEW DISK"
 120 GOSUB 1000: REM READ TAPE
 130 BUF=18
 140 FOR TR=3 TO 34
 150 B1= PEEK (4352+56+4*TR):B2=
      PEEK (4352+57+4*TR)
 160 BM=B1*32+B2/8
 170 FOR SE=12 TO 0 STEP -1
 180 IF BM>4095 THEN 240: REM NOT
     SAVED
 190 GOSUB 2000: REM WRITE SECTOR
 200 BUF=BUF+1
 210 IF BUF<TOP THEN 240
 220 GOSUB 1000: REM READ TAPE
 230 BUF=18
 240 BM=( BM*2 ) MOD 8192
 250 NEXT SE
 260 NEXT TR
 270 PRINT "DISK RESTORED"
 280 END
1000 IF NR( RD )=0 THEN RETURN
1010 PRINT "READING ";NR( RD );" BUFFER
     S"
1020 INPUT "START TAPE AND HIT RETURN
     ",C$
1030 POKE 60,0: POKE 61,18: POKE
     62,255: POKE 63,NR( RD )+17: CALL
     -259
1040 PRINT "STOP TAPE"
1050 RD=RD+1
1060 RETURN
2000 POKE 4118,TR: POKE 4119,SE:
      POKE 4123,BUF: POKE 4113,0

2010 PRINT "TR=";TR;" SE=";SE
2020 CALL 4096
2030 IF PEEK (4113)<>0 THEN PRINT
     "ERROR", PEEK (4113)
2040 RETURN
```

```
3000 NS=0
3010 FOR TR=3 TO 34
3020 B1= PEEK (4352+56+4*TR):B2=
     PEEK (4352+57+4*TR)
3030 BM=B1*32+B2/8
3040 FOR SE=0 TO 12
3050 IF BM<4096 THEN NS=NS+1
3060 BM=(BM*2) MOD 8192
3070 NEXT SE
3080 NEXT TR
3090 NR(0)=0:NR(1)=0:NR(2)=0:NR(
     3)=0
3100 NF=NS/MAX
3110 IF NF=0 THEN 3130
3120 FOR I=0 TO NF-1:NR(I)=MAX: NEXT
     I
3130 NR(NF)=NS MOD MAX
3140 RETURN
10080 REM ************************
10090 REM *  SAVE TAPE RESTORE   *
10100 REM * BY DANA J. SCHWARTZ *
10110 REM *       6/1/80        *
10120 REM ************************
```

A TECHNIQUE TO RUN PROGRAMS FROM AN EXEC OR FROM THE KEYBOARD

John L. Moon

Here is a suggestion about how to code programs so that they can be run either from an Exec or directly from the keyboard - even when some parameters have to be passed to the program.

This might occur with any kind of program, in my case it was the abbs that I was writing. At times, I wanted to invoke the program from the keyboard and have all controls entered by me at that time. I also wanted another mode where the program would be run using a set of default parameters (that I might wish to change from time to time) as it booted from disk. The solution was to program like this:

```
  10 IF ZZ=1 THEN GOTO 100
  20 REM ALL INITIALIZATIONS
     FOLLOW HERE, QUERYING
     THE KEYBOARD, 30 IS AN EXAMPLE
  30 INPUT "FILENAME=";F$
 100 REM REST OF INITIALIZATIONS
```

When run from the keyboard using the RUN command, the variable ZZ is set to 0 by Basic so that statement 30 in the above example would be executed.

To run from an Exec, the Exec would contain lines like:

```
LOAD PROGRAM
ZZ=1
DIM F$(20) (NOTE: INTEGER BASIC ONLY
F$="DEFAULTFILENAME"
GOTO 10
```

In this case, since the program is started with a GOTO statement instead of a RUN statement, the variables are not initialized and the program will detect that ZZ HAS BEEN set and skip over the part of the program that queries the keyboard.

This works best in Applesoft because character variables do not have to be dimensioned first. But it can also be used in Integer Basic if you take care to dimension the character variables correctly.

Dealer's Corner

Writing Interactive Programs

Paul A. Sand
Computerland/Rockville

This article concerns a pet peeve of mine and the Achilles' Heel of many programs: lack of concern for program users by program authors. This lack of concern shows up in the parts of the program that give the user information and expect him to type responses on the keyboard. Far too many programs assume that the user will never make a mistake in entering data from the keyboard. There are few things more frustrating to a computer user than crashing a program by making a simple typing mistake, especially when the crash causes valuable data to be lost or hours of work to be wasted. Many programs fail to communicate to the user just what is expected of him and what his options are. This may cause a time-consuming search of the program documentation (if it exists) or random and dangerous experimentation by the user as he tries to find the magic words the program expects him to type. In short, too many programs are just plain difficult to use, requiring too much effort from the user than necessary. After all, one purpose of the computer is to reduce drudgery; so why are there so many "lazy" programs that ask for extra work from the user?

Needless to say, a program that is difficult and frustrating to use will not be used much at all. Even if it uses the fastest algorithms and provides more features than any similar program, unless it is pleasant and conceptually simple very few people will want to attempt to figure out how to run it succesfully. And an unused program was a waste of time to write in the first place. The following tips may help programmers to write programs that will be "nicer" to the people that use them. Even if the programmer is the only one to ever use his programs, these suggestions will still be worthwhile; in a few months or weeks even he may not be able to figure out his poorly designed program.

The prime rule to remember when writing any interactive code is simply: CHECK ALL INPUT. To state it more strongly: Assume the program's user will make dreadful blunders in entering data, because (by Murphy's law) he is certain to do so. Your program should be protected from the user and the user protected from your program. Although it's impossible to catch all errors, careful input checking will eliminate all but dumb or malicious mistakes.

There are many ways input data entered by the user can be checked, depending on its nature. If the user was to have entered a number, the program should make sure a numeric value was actually typed in. (Some Basic interpreters catch this error.) It is nearly always appropriate to check a numeric input to insure that it lies within a certain range. A too-large number, if not caught, can cause any number of things to happen to a program, all of them bad.

It may also be appropriate to check a numeric input for plausibility. A wrongly entered number may be acceptable in that it doesn't cause the program to crash, but should still be checked carefully; a classic story involves a property tax estimation program that valued a 1967 Ford at over seven million dollars without blinking an eye.

Non-numeric data can also be checked in a number of ways depending on the particular application. It may be that an entered string is too long or short to be processed correctly. It may contain characters that don't make sense. (A person's name shouldn't contain any digits. On the other hand, his Social Security Number shouldn't have anything but digits and dashes.)

When a relatively small number of user responses are considered valid, the program should check that one of those responses was actually entered by the user. The most common example occurs when a program expects a "YES" or "NO" (or "Y" or "N") response; the program should reject any other response. It is truly amazing how many programs fail to do this, assuming that any input that isn't a "Y" is a "N" or vice versa. In just a few minutes of browsing through the July 1980 issue of Creative Computing, I found this failing in four different program listings.

What to do when an input error is detected is a more complicated question. Usually the most appropriate action is to simply re-prompt the user and let him try again. If the programmer wants to make the user feel bad - he shouldn't - a snide error message can be generated. ("Enter a "YES" or "NO" please.") In a few cases, the user may be trying to escape from a piece of the program he wandered into by accident. ("I don't want to enter this record, after all.") and the program could interpret a certain character as a signal by the user to give up on the current process and return to the next higher program level.

A problem related to error checking is how to make the program give the user enough information to answer the questions the computer asks. Many programs don't let the user know what the program expects from them. Of course, this is the whole idea of some game programs such as Adventure. But it isn't very much fun for the user to be forced to figure out the magic words for a more serious program. Although rigorous error-checking should protect users from most of their mistakes, an adequate amount of "prompting" information can save them from making all but pure typing blunders in the first place.

A useful standard for those questions that have a small number of valid responses is simply to list the options available in the prompt line:

```
Do you want to continue?(Y/N):

or

Slow, medium, or fast monsters?(S/M/F):
```

In the case where a range of input values is acceptable, the prompt can also clearly indicate it:

```
What is your height in inches?(36-84):

or

What is the current month?(1-12):
```

Another good method for indicating the user's possible choices is the "menu" - a list of briefly described options from which the user can choose by simply typing a single number or letter. This method is often used for the main control structure of larger programs such as Apple's Controller or File Cabinet.

Default answers can be a boon for poor typists. (A default is the answer assumed by the program to be meant by the user if he simply hits "RETURN" in response to a prompt.) Although handy, defaults should be used with care; the default answer to: "Do you want to delete the data file?" should be NO, not YES! But to use a default, the user must know what the default is. There are a number of possible ways to do this; one is suggested by Bruce Field in the April 1980 Apple Pi (Programming Quickie, page 7), putting the default in inverse video when it appears in a list of legal answers. Another semi-standard (clumsy to do in Apple Basic) is to put the default in square brackets on the prompt line:

```
Enter character size in inches[0.14]:

or

Do you want to continue?(Y/N)[Y]:
```

Still another way is to display the default where the user would type his answer. If the user enters a non-default response, it replaces the default on the screen.

Many programs that use random-access disk files understandably require a restriction on the maximum length of an entered string. If such a requirement exists, the user should be made aware of it. Again, there are a number of methods:

```
Enter Address(25 characters max.):
```

This is very undesirable. Forcing a user to count characters is bound to start him wondering why he got a computer in the first place. Actually he should be wondering why he is using a program that makes him do the dirty work.

A better method is to indicate the maximum length directly on the screen somehow:

```
What is today's date?:........
```

In this case, the eight periods show that the program expects the user to enter no more than eight characters. The user's input replaces the periods as he types it in.

Perversely, although adequate information is necessary for users to input data properly, too much information displayed onscreen is nearly as bad as too little; users will tend not to read anything longer than a few lines. Gabby explanations are generally unappreciated by the person who is learning the program and are useless for the person who has mastered it. They should be saved for the program documentation. Some programs have a "verbose" mode, turned off or on by the user as necessary, that gives more informative error and help messages. In some cases, this mode can almost act as the program's user manual.

By now the reader may be asking: How in the world can I do all this checking and verifying stuff every time my program requests input from the user? If the program expects a lot of such input, an general subroutine is indicated, probably resulting in a net code saving. Isolating input into a subroutine also has the advantage of insuring uniform rules for entering data throughout the program. Here are two such subroutines, one in Applesoft Basic, the other in UCSD Pascal.

The Applesoft routine incorporates many features discussed above. It expects the following variables to be defined before it is called:

yp% - line number on which question is to be asked. If yp% is not in the range 1-24, the current line is used.

xp% - column number where prompt is to begin. It should be small enough so that both the prompt and the response will fit on the line. If xp% is not in the range 1 to (39 - len(qu$) - ml%) then the prompt will start in the first column.

qu$ - prompting question.

df$ - default answer. The default will appear after the prompt, as discussed above. if df$ = null then no default is accepted.

va$ - string containing "legal" input characters. Any character typed in that is not contained in va$ will not be accepted. If va$ = null then such character checking is not done.

ml% - maximum length of input string. A string of ml% periods will be displayed after the prompt as discussed above, although some or all of it may be overwritten by the default. Attempts to enter characters beyond the maximum length won't work. Also, the output answer will be padded on the right with blanks, if necessary, to a total length of ml% characters. If ml% = 0 then none of this is done.

mn$, mx$ - minimum and maximum strings, respectively. They are used for range checking; if the answer is not between mn$ and mx$ inclusive (using Applesoft string comparison), the user is reprompted. If mn$ >= mx$ then no range checking is done.

The user's answer is returned in the variable an$. Since a lot of string manipulation is done within the subroutine, a freespace calculation is done at the end to force Applesoft to clean house. A typical yes/no answer would be obtained as follows:

```
100 qu$ = "Do you want to continue?(Y/N)"
110 mn$ = "": mx$ = "": yp% = 12: xp%=10
120 df$ = "Y": va$ = "YN": ml% = 1
130 gosub 10000
```

Or a zip code:

```
200 qu$ = "Zip Code?": mn$ = "00000":mx$ = "99999"
210 yp% = 20: xp% = 1: df$ = ""
220 va$ = "0123456789": ml% = 5
230 gosub 10000
```

The Pascal function is less complex to use. It picks a default answer and a list of possible answers out of the prompt question string. The prompt should be in the form:

```
Question?(pos1/pos2/.../posN)[default]:
```

where both the default and list of possible answers are optional. If a default is given and the user simply hits "return" then the default is returned to the calling routine. If no list of possible answers is given, the function returns whatever the user typed in, or the default. If the list is

given, then the user's answer is checked to insure that he typed one of them.

Since UCSD Pascal (unfortunately) does not allow strings to be returned as function values, the user's answer is returned as a var parameter in the argument list. However, the first character of the response is returned by the function, allowing straightforward code like:

```
repeat
   playagame
until (answer('Want to play again?(Y/N)[Y]:', ans) = 'N');
```

or

```
case answer('What now?(ADD/CHANGE/DELETE/EXIT):', ans) of
   'A': addrec;
   'C': changerec;
   'D': deleterec;
   'E': exit(program)
end;
```

These suggestions are certainly not meant to be the final answers to the problem of how to handle the interaction between the user and the computer. Every program will have its own requirements for input checking and user prompting. But the basic lesson remains: programmers should give the users of their programs every chance to run them pleasantly and succesfully.

```
rem --------------------------------
rem General Interactive Input Routine
rem --------------------------------

10000 if yp% > 0 and yp% < 25 then vtab yp%
10010 if xp% < 1 or xp% > 39 - len(qu$) - ml% then xp% = 1
10020 htab xp%: print qu$;":";
10030 if ml% <= 0 then 10050
10040 for i = 1 to ml%: print ".";: next i
10050 an$ = "": xc% = xp% + len(qu$) + 1
10060 htab xc%: print df$;: htab xc%
10070 get ch$: ch% = asc(ch$): if ch% = 13 then 10220
10080 if ch% <> 8 then 10160
10090 if len(an$) <= 0 then print chr$(7);: go to 10070
10100 xc% = xc% - 1: htab xc%
10110 if ml% <= 0 then print " ";: go to 10130
10120 print ".";
10130 htab xc%
10140 if len(an$) <= 1 then an$ = "": go to 10070
10150 an$ = left$(an$, len(an$) - 1): go to 10070
10160 if va$ = "" then 10200
10170 i% = 1
10180 if i% > len(va$) then print chr$(7);: go to 10070
10190 if mid$(va$, i%, 1) <> ch$ then i% = i% + 1: go to 10180
10200 if len(an$) >= ml% and ml% > 0 then print chr$(7);: go to 10070
10210 htab xc%: print ch$;: xc% = xc% + 1: an$ = an$ + ch$: go to 10070
10220 if an$ = "" then an$ = df$
10230 if an$ = "" then print chr$(7);: go to 10000
10240 if len(an$) < ml% and ml% > 0 then an$ = an$ + " ": go to 10240
10250 if mn$ >= mx$ or (an$ >= mn$ and an$ <= mx$) then 10270
10260 print chr$(7);: go to 10000
10270 htab xp% + len(qu$) + 1: print an$;
10280 i = fre(0)
10290 return
```



```
function answer(question: string; var ans: string): char;

const
   MAXOPT = 10;
   M1 = 11; (* MAXOPT + 1 *)

var
   default, response, temp: string;
   p1, p2, nopts, i: integer;
   opt: array [1..M1] of string;

begin
   p1 := pos('[', question);
   p2 := pos(']', question);
   if (p1 > 0) and (p2 > 0) and (p2 > p1 + 1) then
      default := copy(question, p1 + 1, p2 - p1 - 1)
   else
      default := '';
   nopts := 0;
   temp := question;
   p1 := pos('(', temp);
   if p1 > 0 then begin
      delete(temp, 1, p1);
      p2 := pos('/', temp);
      while (p2 > 1) and (nopts < MAXOPT) do begin
         nopts := nopts + 1;
         opt[nopts] := copy(temp, 1, p2 - 1);
         delete(temp, 1, p2);
         p2 := pos('/', temp)
      end;
      p2 := pos(')', temp);
      if (p2 > 1) and (nopts < MAXOPT) then begin
         nopts := nopts + 1;
         opt[nopts] := copy(temp, 1, p2 - 1)
      end
      else (* bad syntax in question *)
         nopts := 0
   end;
   repeat
      write(question);
      readln(response);
      ans := '';
      if (response = '') and (default <> '') then
         ans := default
      else if nopts <= 1 then
         ans := response
      else begin
         opt[nopts + 1] := response;
         i := 1;
         while opt[i] <> response do
            i := i + 1;
         if i <= nopts then
            ans := opt[i]
      end
   until ans <> '';
   answer := ans[1]
end;
```

*** WASHINGTON APPLE DIGEST ***

CREATIVE COMPUTING, August 1980

Visicalc product review, P26. Good summary of capabilities.

Computer Bismarck, product review, P31. Very sophisticated rules, and at least several hours must be spent reading them before initiating the game. Games can be saved to disk after any turn. As an alternative, your computer can assume the role of Otto, the simulated admiral of the opposition forces. (Strategic Simulations, Inc., $59.95).

Sorting techniques, P35. Part I of a three-part series. Good technical discussion of the "insertion sort" method. You have to think about it while you read the article.

Word processing, P38. Review of the "Magic Wand" software product. This is a package which sells for $400 and does not apply to Apple computers, but it describes a number of features of word processing that might make the article of interest to some.

Marketing your own software, P52. This article concludes that there is a lot of psychological reward, but little money. It gives some hints for those who are interested in selling their creations.

Copyrighting, P140. Good discussion of the laws and implications.

Stock market analysis, P56. Part 4 of a description of a software product for analysis. It's a long article, and it might be useful to those interested. A version of the program is available for the Apple.

Graphics, P110. Plotting of pie-charts, program and discussion. P116: Plotting with the Diablo printer.

Apple products, P148. Disk speed adjustment; another review of the Apple III, including SOS, mail list manager, business Basic.

Games, P142. An article on the theory of writing game software.

Robots, P74. Review of a conference on the subject.

INFOWORLD, July 21, 1980

Small business computer systems general ledger, P12. The author summarizes as follows: "I heartily recommend this general ledger for those who only need a general ledger for their Apple and who are familiar enough with computerized business systems so that documentation won't overwhelm them. For the average business man or the computer novice, however, I would say that they should investigate Apple's Controller package or wait until the rest of the Osborne programs have been converted and better end-user manuals created." A fair review. Should be read before purchase (or development) of any business-oriented software.

DOS 3.3 (and DOS toolkit), P15. Elementary. Announcement of DOS 3.3 and the DOS Toolkit, a set of software tools that lets Applesoft people write more easily in assembly, costing $75.00. If you don't have a language card (which lets you use a real assembler), the Toolkit looks interesting.

Videotape meets Micro, P50. Elementary. Announcement of "an intelligent VTR (Video Tape Recorder) controller to integrate an Apple II with a video player." This is not another Winchester backup; rather it is for intelligent CAI (Computer Aided Instruction). This reminds me of the old film strip projectors hooked up to tape recorders. They have their place, but does the world really need them?

INFOWORLD, August 4, 1980

According to Garetz, P12. Elementary(Rumor) A well-informed source inside Apple Computers says that one of Apple's next-generation machines (beyond Apple III) is being designed in a unique way. Apple plans to write the user manuals before designing any hardware or software. If Apple can pull it off, the highly useroriented machine should be the ideal personal computer that only visionaries dream of. This source also comments that Apple has the resources to succeed. There seems to be inconsistencies in this article.

Testing students a better way, P12. Software review (intermediate level). "Aristotle's Apple" (Stoneware Microcomputer Products) is a generalized, teacher-oriented authoring program that can be used to construct multiple choice, fill-in-the-blank, and column matching drills. Aristotle, as the Stoneware people refer to the package, also contains a file editing system for changing and deleting drills, and allows the drills to be presented in a "tutor" of "exam" mode. The review goes on to evaluate the programs's functionality (fair), ease of use (good), documentation (poor), error handling (good), and support (good). It is fairly lengthy and detailed, and supports its criticisms well. If I were to be in the market for computer-aided instruction tools, I would use this article as a yardstick for evaluating other packages as well.

Disk Apple II Report Textwriter (DART), P15 Elementary. The DART processing software was designed to simplify the creation of letters, reports, and other output text. Options include pagination, line justification, titles/subtitles, and page numbering ($19.95). Cheap, interesting, but don't these things exist in profusion now-a-days?

BYTE, July 1980

The theme of the issue is computers in education. Two items are for or about Apple. Several are of general interest.

Apple III product description, P50. Based on the Apple product description available at most dealers. It includes some color pictures of the Apple III and its high resolution graphics display.

Interactive control of a video cassette recorder with a personal computer, P116. For hardware/software experts. Explains how to control a Sony SLO-320 video recorder with your Apple. Includes schematic for an interface board (which you would have to build) and listings of Basic programs and assembler routines.

Pilot/P: Implementing a high-level language in a hurry, P154. For Pascal programmers. Describes a pre-processor which translates PILOT programs into Pascal programs. Includes a complete listing of the translator -- about 160 lines of UCSD Pascal. Includes two short sample PILOT programs, otherwise assumes you know how to use PILOT.

Simulating human decision making, P56. A well-written article about game playing strategies using the game of Othello as an example. Learn how to beat your computer or invent smart programs.

Creating a fantasy world on the 8080, P210. For ambitious game programmers. It describes advanced programming techniques used in adventure type programs.

BYTE, August 1980

The theme of this issue is the FORTH language. No articles are particularly aimed at Apple owners. This might be an issue to skip unless you are thinking about taking a plunge into FORTH programming. If you are interested in FORTH, the articles will give you some idea of how the language works. The articles are a bit too technical for the novice and probably too simple for experts.

- - - - - - - - - -

Editor's note

Thanks to these contributing editors: Dave Efron, Mike Leavitt, and Bill Schultheis.

We're still searching for more contributors to cover a broader representation of information sources. We're also searching for a standard format for publishing our reviews. Call 251-0225 or see Dave Efron at the meeting if you are interested in helping.

As a suggestion, it would be ideal to have several persons covering sources like Creative Computing. One individual could scan and review articles dealing specifically with games, while another person could cover graphics, etc. Some of these topic areas require a reviewer with some special knowledge or interest in the subject. For example, a more extensive review of games and articles about games would be performed by someone with an interest in the subject.

As time progresses perhaps interest in this column will afford broader and more extensive coverage.

ODYSSEY- THE COMPLETE ADVENTURE

Perhaps the best way to begin this review is to relate what I remember about the author, Bob Clardy, from a telephone conversation with him about a year ago. Before moving to Washington State from California, Bob was (and possibly still is) a "Dungeons and Dragons" fanatic. In Washington, Bob bought an APPLE II. His first commercial effort was entitled "Dungeon Campaign", a multi-level dungeon maze with several innovations in the use of real-time monsters and user interaction. Then followed "Wilderness Campaign" which, until now, I had considered to be the finest blend of APPLE and "D&D". Wilderness introduced Hi-Res into APPLE fantasy gaming. Moreso, a Wilderness game was a scenario and a player had the impression of living a story. Unfortunately, Wilderness had a few bugs and these bugs were what lead me to call Bob (and learn the preceding in the progress). Within days, I received an errata sheet in the mail and all became perfect.

In the year that followed, I'd seen Bob's name on some utility and commercial type programs and had assumed he had forsaken fantasy programming. Then, just a few days ago, I spotted Odyssey at a local computer store. As soon as I spotted the author's name, I knew that I need look no further to determine if the program was worth its price, and so, the following review is from a very satisfied customer.

Odyssey is quite extensive. There is so much programming on the diskette that it was necessary to remove the DOS to make room for everything. The user's first instruction is to "boot" DOS with any standard 3.2 diskette. APPLE PLUS users beware that Odyssey requires Integer Basic and 48K in order to run. Although a 16 page instruction manual is provided, several readings are necessary and (probably) several tries of the game also before Odyssey is understood. This is in no way a detraction and merely typifies the extensiveness of the game. A full Odyssey takes several hours and the author thoughtfully provided a "SAVE GAME" feature.

Odyssey is a role-playing game with more "D&D" elements than I would have believed possible. The user will experience many "Adventure" type situations and "dungeon" type situations in new formats. The basic scenario is to accumulate necessary force to topple the evil ruler, but that is a gross oversimplification. A typical (but condensed) game follows.

At the beginning the player starts on a hi-res map of an island near a village. The first step is to go to the village and buy food/weapons/armor/supplies necessary to search the island to build a large enough force. Barter with the merchants and the price might come down, but if you barter too much the merchant might refuse to sell. Now search the island. Various treasures are there to find with the right resources. Unless you have a light source, you will not be able to search the dark ruins. (Adventure elements are used extensively). Various obstacles can not be overcome without the correct supplies. You will search jungle, swamp, and mountains for supplies and treasure. You will be attacked by all sorts of characters. Mercenaries may join your force or attack you. You may be attacked or offered assistance by magical people. You never know if the "old hag" is offering you a bargain of trying to gyp you.

As you recruit your force, you incur obligations. The larger the force, the more money you need to feed and arm them. Acquire more than your force can carry and you will have to drop items, but be forwarned, virtually everything you acquire has some use somewhere in the game. That fruit you may have passed up purchasing saves your force from scurvy at sea. Finally you acquire enough money and resources (if you don't get killed in the effort) and you go to the port city and buy a ship. Be sure you have everything you might need, for once you leave the original island, you are not allowed to return.

After you answer the question "Do you wish to leave the island" with a yes, the map of the island disappears and is replaced by a hi-res sea map. Not to pun, but all is not smooth sailing. If you had bought the right equipment you will be aware of the wind and current factors. You need these in order to manage your sails and navigate to another island. Lose control and you may sail off the earth. Watch out for pirates, sea monsters and flying monsters. If you planned wisely, all may be overcome with the right assets, otherwise you will suffer losses. You will experience bad weather conditions such as fogs, calms, and storms. All of these cause problems of one sort or another. WATCH OUT FOR THE WHIRLPOOL!

Survive long enough and you land on another island. If it is the island of Caliph,leave quickly, for you will be wiped out without the Magic Orb.(The orb is normally found in the catacombs under the Temple of Mordil). If it is an empty island, restock food and water and set sail for another island. Finally you land at the temple and the wizard at the entrance will not let you pass.

Somehow you find the right way to pass the wizard and you find yourself in lo-res caverns. As you search the caverns for the orb, you also run into several hazards. After you find the orb, you must also find a way out.

All has been successful, you found the orb, departed (the sea map is back), and found your way to the island of the Caliph.(Another hi-res map). You approach the castle and experience new hazards and traps. Suddenly some of those things you acquired along the trip are useful in passing the hazards. (Hint: If you remember your mythology and you know how Perseus killed the Medusa, be prepared to do the same). But be aware that the author randomized these final hazards. In several games, you may face entirely different puzzles requiring the right resources.

Finally, you get to the castle and topple the Caliph and the game is over. Final statistics are displayed along with a composite score. My two sons are still trying to beat the 75,340 that I set (and so am I).

Even after playing the game, there is still enjoyment in trying to better results on a replay. Flex your faculties with fun, fantasy, and frustration. Try Odyssey.

Product is available for $30 from Synergistic Software, 5221 120th Ave, s.e. Bellevue, wa 98006.

---------- Sandy Greenfarb

*** BIORHYTHMS ***

THIS IS A LONG PROGRAM, AND FOR THAT REASON, I (BLUSH) TO SEND IT IN. IT IS ALSO QUITE FAST AND VERSATILE, THE GRAPHS ARE NICELY CENTERED AND ACCURATE, AND DATES AS EARLY AS MARCH 1, 1700 ARE ACCOMMODATED.

CALENDAR AND DAY-OF-THE-WEEK CALCULATIONS HAVE BEEN FASCINATING TO ME FOR SOME TIME, ESPECIALLY AFTER SEEING JOHN L. MOON'S 'PERPETUAL CALENDAR', IN THE SEPTEMBER, 1979 ISSUE OF WASHINGTON APPLE PI NEWSLETTER. THE PROCEDURES IN THIS PROGRAM WERE OBTAINED FROM ONE OF MY HEWLETT-PACKARD CALCULATOR MANUALS, AND WORK VERY WELL, INDEED. (I WAS CURIOUS ABOUT THE '30.6' CONSTANT -LINE 520- AND FINALLY DISCOVERED THAT IT IS THE AVERAGE NUMBER OF DAYS PER MONTH FOR THE 11 MONTHS EXCLUDING FEBRUARY.)
VERY INTERESTING!

I HAVE INCLUDED AN OPTION WHICH I HAVE NEVER HEARD OF BEFORE, BUT WHICH MAY HAVE SOME VALIDITY:

)))--) A FOURTH CURVE (SUMMATION OF THE THREE 'USUAL' BIORHYTHM CURVES) CAN BE PLOTTED, AND MAY INDICATE OPTIMUM TIMES FOR ACTIVITY (OR REVITALIZATION).

ANYWAY, HERE IT IS FOR WHAT IT'S WORTH.

SINCERELY,

HOWIE MITCHELL
7823 SW. 55TH PLACE
GAINESVILLE, FLA. 32601
AUGUST, 1980

```
5 SLOT = 1
10 REM ************************
11 REM *                      *
12 REM * A N  E L E G A N T  *
13 REM * BIORHYTHM PROGRAM    *
14 REM *                      *
15 REM * FOR: THE LRC 7000+   *
16 REM * PRINTER (EATON CO.)  *
17 REM *                      *
18 REM * H.MITCHELL: JULY'80  *
19 REM *                      *
20 REM ************************
50 TEXT : HOME
100 REM ************************                    **
INITIALIZE: DIMENSION, ** SET STRINGS, DATA FOR  ** # DAYS I
N MONTH.      **                    ********************
*********
105 DIM MO$(12),DM(12)
110 DATA JANUARY,31,FEBRUARY,28,MARCH,31,APRIL,30,MAY,31,JUNE,3
0,JULY,31,AUGUST,31,SEPTEMBER,30,OCTOBER,31,NOVEMBER,30,DECEMBER
,31
115 FOR N = 1 TO 12: READ MO$(N),DM(N): NEXT
120 DATA SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURD
AY,SUNDAY
125 FOR N = 0 TO 7: READ DA$(N): NEXT N
130 REM ************************                    **
MO$(N) = MONTH NAME     **  DM (N) = # DAYS IN M'TH.**  DA$(N) =
 DAY NAMES.    **                    ********************
*********
135 PR$(1) = CHR$ (4) + "PR#" +  STR$ (SLOT) +  CHR$ (13) +  CH
R$ (15) +  CHR$ (31):PR$(0) =  CHR$ (4) + "PR#0" +  CHR$ (13)
140 REM ************************                    **
PR$(1): PRINTER ON      **  (@ 64 CHAR/LINE)      ** PR$(0):P
RINTER OFF     **  (@ 0 CHAR/LINE).      **
      ******************************
200 REM ************************                    **
PROGRAM TITLE & INPUT  **                    ************
****************
204 TEXT : HOME
208 HTAB 3
212 PRINT "*** ELEGANT BIORHYTHM PROGRAM ***          -------
----------": PRINT
216 PRINT "  THIS PROGRAM WILL DRAW GRAPHS OF YOUR BIORHYTHM CY
CLES, FOR A 3-MONTH PERIOD, BEGINNING ON THE FIRST OF ANY SPECIF
IED MONTH AND YEAR.": PRINT
220 PRINT "  IN ADDITION, IT WILL PRINT OUT SOME  BASIC INFORM
ATION (IF YOU WISH) AS TO  WHAT BIORHYTHMS ARE."
224 PRINT : PRINT : INPUT "((( PRESS RETURN )))";HOLD$: HOME
228 PRINT "  PLEASE SELECT FROM THE FOLLOWING:      ---------
--------------------
232 PRINT "  1. BASIC BIO-CYCLE GRAPH.": PRINT : PRINT "  2. BA
SIC BIO-GRAPHS + INFORMATION.": HTAB 6: PRINT "(ON WHAT BIORHYTH
MS ARE, ETC.)": PRINT
236 PRINT "  3. BASIC BIO-GRAPHS + 'SUM-WAVE'          (SUMMAT
ION CURVE FOR THE THREE         CYCLES ADDED TOGETHER).
240 PRINT : PRINT "  4. BIO-GRAPHS+SUM WAVE+INFORMATION.   "
244 PRINT "  5. PRINT DATA FROM DATA LINES.": HTAB 6: PRINT "(U
SE LINES #450-498).": PRINT : PRINT "  6. AUTOMATIC PRINTOUT OF
A SERIES OF       CHARTS, FROM DATA LINES.": PRINT
248 PRINT "  7. END      8. END & DELETE DATA.": PRINT
248 PRINT "  WHICH DO YOU CHOOSE ?";: GET CHOICE$:CHOICE =  VAL
 (CHOICE$)
252 PRINT "** CHOICE #";CH;" **": IF CH ( 1 OR CH ) 6 THEN  PRI
NT "  (PROGRAM ENDED.)";
253 IF CH = 7 THEN  VTAB 23: END
254 IF CH = 8 THEN  PRINT " (DATA DELETED)";: VTAB 23: DEL 450,
498: END
255 IF CHOICE ( 1 OR CHOICE ) 8 THEN  VTAB 23: END
256 FOR N = 0 TO 1000: NEXT : HOME : IF CHOICE = 5 THEN 400
258 IF CHOICE = 6 THEN 600
260 INPUT "  PLEASE TYPE IN YOUR FIRST NAME, COMMA,  THEN YOUR
LAST NAME:
))--) ";NA$(1),NA$(2)
262 UNO = 1: REM :FLAG FOR SINGLE ENTRY.
264 PRINT : PRINT "  THANK YOU, ";NA$(1);".": PRINT
268 INPUT "  NOW, PLEASE TYPE IN YOUR BIRTHDATE   (MONTH, DAY,
 YEAR), SEPARATING THE      THREE NUMBERS WITH COMMAS:
))--) ";BM,BD,BY
272 IF BY ( 100 THEN BY = BY + 1900
276 GOSUB 500:BD$(1) = BD$: REM :GET JULIAN DATE,D'DAY OF WEEK.
```

```
280  PRINT : PRINT "### JULIAN DAY NO. = ";JD;" ###": PRINT : PR
INT NA$(1);", YOU WERE BORN ON A ";BDAY$;"."
284  PRINT : PRINT "  NEXT, PLEASE TYPE IN THE MONTH AND    YEAR
 FOR THE BEGINNING OF YOUR CHART:": PRINT
286  INPUT "))--) (MM,YY) ? ";PM,PY: IF PY ( 100 THEN PY = PY +
1900
288 LIVE = JD:Y = PY:M = PM:D = 1: GOSUB 505:LIVE =  INT (JD - L
IVE): REM : NOW, "LIVE" GIVES # DAYS PERSON HAS BEEN ALIVE AS OF
 THE BEGINNING OF THE FIRST PREDICTION MONTH.
290  PRINT : PRINT "### JULIAN DAY NUMBER = ";JD;" ###": PRINT :
 PRINT "DIFFERENCE = ";LIVE;" DAYS."
292  PRINT : PRINT "  GOOD, AND THANK YOU, ";NA$(1);".": PRINT
294  PRINT "  I NOW HAVE ENOUGH DATA TO PREPARE     YOUR CHART.
 YOU SEE, BY KNOWING THE    JULIAN DAY DIFFERENCE ("LIVE;"), AN
I"
296  PRINT "BY ASSUMING THAT THE 3 CYCLES BEGIN TO- GETHER ON YO
UR BIRTHDATE, I CAN CALCU-  LATE THE STAGE OF EACH CYCLE AT THE
BE- GINNING OF ";MO$(PM);",";PY;", AND PROGRESS"
298  PRINT "THEM DAY-BY-DAY FROM THERE.": PRINT
299  INPUT "((( PRESS RETURN ))) ";HOLD$: HOME
399  GOSUB 650: END
400  REM ************************                              **
ROUTINE TO PRINT OUT    **  DATA LINE DETAILS:      **
            ********************************
403  TEXT : HOME
406  READ NAME$(1): IF NAME$(1) (  ) "(END OF DATA)" THEN 409
407  PRINT  CHR$ (7): PRINT NAME$(1): PRINT : PRINT "  SORRY. OU
T OF DATA IN LINES #450-498.": PRINT
408  PRINT "  ENTER DATA IN LINES #450-498 , USING    THE FOLLOW
ING FORMAT:": PRINT : PRINT "LINE# DATA FIRST NAME,LAST NAME,BR
TH M,BRTH DAY,BRTH YR,PROJ.MO,PROJ.YR,CHOICE#": END
409  RESTORE : FOR N = 1 TO 32: READ A$: NEXT : REM : READ UP TH
ROUGH 12 MONTH & 12 DAYS-OF-MONTH STRINGS (24) + SUNDAY THRU SUN
DAY (8 MORE).
412  REM ************************ PRINT DATA ENTRY DETAILS ****
**************************
415  PRINT PR$(1)
418  PRINT  CHR$ (30): HTAB 11: PRINT "DATA LINE ENTRIES:": PRIN
T  CHR$ (31)
421  PRINT "  (NAME)          (BIRTHDATE)     (CHOICE)   (P
RED.DATE)": FOR N = 1 TO 63: PRINT "-";: NEXT N: PRINT
424  READ NAME$(1): IF NAME$(1) = "(END OF DATA)" THEN  FOR N =
 1 TO 63: PRINT "-";: NEXT : PRINT : PRINT PR$(0): END
427  READ NAME$(2),BM,BD,BY,PM,PY,CHOICE: IF BY ( 100 THEN BY =
BY + 1900
430  IF PY ( 100 THEN PY = PY + 1900
433  GOSUB 500: REM : JULIAN CALC. & B'DAY OF WEEK.
436 LINE$ = NA$(1) + " " + NA$(2): FOR N =  LEN (LI$) TO 18:LI$
= LI$ + " ": NEXT :LI$ = LI$ + BDAY$ + " " +  STR$ (BM) + "/" +
 STR$ (BD) + "/" +  STR$ (BY): FOR N =  LEN (LI$) TO 41:LI$ = LI
$ + " ": NEXT :LI$ = LI$ +  STR$ (CHOICE)
439  FOR N =  LEN (LI$) TO 49:LI$ = LI$ + " ": NEXT :LI$ = LI$ +
 MO$(PM) + " " +  STR$ (PY)
442  PRINT LI$
445  GOTO 424
449  REM ************************                              **
USE LINES #450-498 FOR  **  DATA. FORMAT:LINE#DATA **  FNAME,LN
AME,BM,BD,BY,   **  PM,PY,CHOICE.          **
      ********************************
499  END
500  REM ************************                              **
JULIAN DAY NUMBER, AND  **  BIRTH DAY-OF-THE-WEEK.  **
            ********************************
503 Y = BY:M = BM:D = BD
505  IF Y / 4 =  INT (Y / 4) THEN DM(2) = 29
506  IF Y / 4 (  ) INT (Y / 4) THEN DM(2) = 28: REM : CORRECT D
AYS-IN-FEBRUARY FOR LEAP-YEAR.
509  IF M ) 2 THEN M = M + 1
510  IF M ( 3 THEN Y = Y - 1:M = M + 13
515  REM ************************                              ** I
N THE NEXT STEP, JD IS  ** THE 'JULIAN DAY NUMBER', ** MEASURED
FROM MAR.1,1700.**                          **********************
*********
520 JD = 365.25 * Y +  INT (30.6 * M) + D - 621049
525 JD = JD + (BY ( 1901 AND M ) 12) + (BY ( 1900) + (BY ( 1801
AND M ) 12) + (BY ( 1800)
530 WEEKDAY =  INT (( INT (JD' / 7 -  INT (JD / 7)) * 7 + .5)
535 BDAY$ = DAY$(WEEKDAY)
540  RETURN
600  REM ************************                              **
AUTOMATIC PRINTOUT OF   **  BIO-CHARTS, FROM DATA   **  STATEMEN
TS.          **                          **********************
*********
605  RESTORE : FOR N = 1 TO 32: READ A$: NEXT N: REM : READ THRO
UGH INITIALIZING DATA, TO "REACH" BIO-DATA.
610  READ NAME$(1): IF NAME$(1) = "(END OF DATA)" THEN  PRINT PR
$(0): PRINT NAME$(1): END : REM : END WHEN BIO-DATA IS EXHAUSTED
.
615  READ NAME$(2),BM,BD,BY,PM,PY,CHOICE
616  IF BY ( 100 THEN BY = BY + 1900
617  IF PY ( 100 THEN PY = PY + 1900
620  GOSUB 500:LIVE = JD:BD$(1) = BD$
625 Y = PY:M = PM:D = 1: GOSUB 505:LIVE =  INT (JD - LIVE)
630 A$ = ""
649  REM ************************                              **
  B I O G R A P H     **                          ************
******************
650 PI = 355 / 113: IF CHOICE = 2 OR CHOICE = 4 THEN  GOSUB 1000
: REM : PRINT OUT INFORMATION.
652  PRINT PR$(1)
655  PRINT : PRINT : PRINT  CHR$ (31); CHR$ (14): REM : 32 CHAR/
LINE
660  HTAB 11: PRINT "BIO-CYCLES": PRINT  CHR$ (15): REM : BACK T
O 64 CHR/LINE.
665  REM ************************                              **
MO$(M) = MONTH NAME     **  DM (N) = # DAYS IN M'TH.**  DA$(N) =
 DAY NAMES.     **                          **********************
*********
670  REM ************************ )INPUT VARIABLES(        ** B
M,BD,BY=BIRTH MO,DA,YR  ** PM,PY=PROJECTION MO,YR   ** M,B,Y=USE
D IN JULIAN DAY ** ROUTINE                  ** LIVE=NO.DAYS LIVE
D,PM,PY ********************************
675 PFRACT = LIVE / 23 -  INT (LIVE / 23):EFRACT = LIVE / 28 -
INT (LIVE / 28):MFRACT = LIVE / 33 -  INT (LIVE / 33): REM GET F
RACTION OF CYCLES FOR  BEGINNING OF CURRENT MONTH.
680 PF = PF * 2 * PI:EF = EF * 2 * PI:MF = MF * 2 * PI: REM : ST
ARTING ANGLES IN RADIANS.
685 DP = 2 * PI / 23:DE = 2 * PI / 28:DM = 2 * PI / 33: REM : DA
ILY ANGLE INCREMENTS IN RADIANS.
687 AMP = 20: IF CHOICE = 3 OR CHOICE = 4 THEN AMP = 15: REM : W
AVE AMPLITUDE (LESS IF SUMMATION-WAVE INCLUDED.)
690  DEF  FN PHYS(A) = 32 +  SIN (PF + COUNT * DP) * AMP
692  DEF  FN EMOT(A) = 32 +  SIN (EF + COUNT * DE) * AMP
694  DEF  FN MENT(A) = 32 +  SIN (MF + COUNT * DM) * AMP
696  DEF  FN SUM(A) =  FN PHYS(A) +  FN MENT(A) +  FN EMOT(A) -
64
```

```
702  PRINT  CHR$ (30): REM : 40 CHR/LINE
705  PRINT "  BIO-CYCLES FOR "¡NA$(1);" "¡NA$(2);":": PRINT
710  PRINT "  (FOR "¡MO$(PM);", "¡PY;" THROUGH "¡MO$(PM + 2 - 12
 * (PM ) 10));")"
715  PRINT : PRINT "BIRTHDATE: "¡BD$(1);", "¡MO$(BM);" "¡BD;", "
¡BY
717  IF CHOICE = 3 OR CHOICE = 4 THEN  PRINT : PRINT ")))—) (SU
N WAVE SHOWN DOTTED) (—(((("
720  PRINT  CHR$ (31): REM : 64 CHR/LINE
722  FOR N = 1 TO 63:SPACE$ = SPACE$ + " ":DASH$ = DASH$ + "-":
NEXT : REM : STRINGS TO DRAW UPON FOR STUFFING P'S, E'S, N'S, ET
C., INTO!
723 COUNT = 0
724 CELEBRASHUN$ = "HAPPY BIRTHDAY, " + NAME$(1) + " ! ":CEL =
LEN (CEL$)
725  FOR CYCLE = 0 TO 2: REM : 3-MONTH PREDICTION LOOP.
730 WHEN = PM + CYCLE - 12 * (PM + CYCLE ) 12):MD = DM(WHEN)
735  FOR SLICE = 1 TO MD
740 PZ =  FN PHYS(A):EZ =  FN EMOT(A):NZ =  FN MENT(A):SUMZ =  F
N SUM(A)
745  IF SLICE = 1 THEN A$ = DASH$
750  IF SLICE ) 1 THEN A$ = SPACE$
755 A$ =  LEFT$ (A$,PZ - 1) + "P" +  RIGHT$ (A$,63 - PZ):A$ =  L
EFT$ (A$,EZ - 1) + "E" +  RIGHT$ (A$,63 - EZ):A$ =  LEFT$ (A$,NZ
 - 1) + "N" +  RIGHT$ (A$,63 - NZ)
760 A$ =  LEFT$ (A$,30) +  STR$ (SLICE) +  RIGHT$ (A$,32 -  LEN
( STR$ (SLICE)))
765  IF CHOICE = 3 OR CHOICE = 4 THEN  IF SUMZ ) 1 AND SUMZ ( 62
 THEN A$ =  LEFT$ (A$,SUMZ - 1) + "." +  RIGHT$ (A$,62 - SUMZ)
767  IF WHEN = BM AND SLICE = BD THEN A$ = CEL$ +  RIGHT$ (A$,62
 - CEL)
770  IF SLICE = 1 THEN L =  LEN (MO$(WHEN)):L = L / 2:A$ =  LEFT
$ (A$,31 - L) + MO$(WHEN) +  RIGHT$ (A$,31 - L): REM : INSERT MO
NTH NAME.
775  PRINT A$
782 COUNT = COUNT + 1: IF SLICE = 1 AND WHEN = 12 THEN Y = PY +
1: GOSUB 505: REM : TRIGGER LEAP-YEAR ROUTINE FOR COMING YEAR.
783  IF  PEEK (37) ) 22 THEN  HOME : REM : BY ERASING THE SCREEN
, THE PRINTER GETS ITS DATA FASTER !
785  NEXT SLICE
790  NEXT CYCLE
805  IF UNO = 1 THEN  PRINT PR$(0): RETURN : REM  : LEAVE GRAPH
890  GOTO 610
900  PRINT PR$(0)
998  END
999  REM ********************************                    ** I
NTRODUCTION & DISCUSSION** OF BIORHYTHMS.          **
                ********************************
1000  PRINT PR$(1); CHR$ (14): FOR X = 1 TO 5: PRINT : NEXT : HT
AB 11: PRINT "BIORHYTHMS": PRINT  CHR$ (15); CHR$ (30): PRINT "
 BIORHYTHMS ARE THOUGHT TO AFFECT BE-  HAVIOR. IT IS ASSUMED THA
T OUR PHYSICAL,MENTAL, AND EMOTIONAL ENERGIES RISE AND"
1010  PRINT "FALL IN RHYTHMIC CYCLES (23-DAY CYCLE  FOR PHYSICA
L,  33-DAY CYCLE FOR MENTAL,"
1020  PRINT "AND 28-DAY CYCLE FOR EMOTIONAL)."
1040  PRINT : PRINT "  WHEN THESE CYCLES CROSS A ZERO 'BASE- LIN
E', THE FUNCTIONS CHANGE PHASE - BE- COME UNSTABLE - AND THIS RE
SULTS IN"
1050  PRINT "SO-CALLED CRITICAL DAYS."
1060  PRINT : PRINT "  THESE 'CRITICAL DAYS' ARE, ACCORDING"
1070  PRINT "TO THE THEORY, OUR WEAKER AND MORE VUL- NERABLE TIM
ES.  ACCIDENTS,  CATCHING"
1080  PRINT "COLDS, AND BODILY HARM ARE  MORE LIKELY ON PHYSICAL
LY-CRITICAL DAYS. DEPRESSION,";
1090  PRINT "QUARRELS, AND FRUSTRATION ARE MORE LIKE-LY ON EMOTI
ONALLY-CRITICAL DAYS.  SLOW-"
1100  PRINT "NESS OF THE MIND, RESISTANCE TO NEW SIT-UATIONS, AN
D UNCLEAR THINKING ARE MORE"
1110  PRINT "LIKELY ON MENTALLY-CRITICAL DAYS.": PRINT PR$(0)
1115  REM ********************                              **
EXAMPLE OF CYCLE GRAPH  **  (COARSE, ON SCREEN)    **
          ********************************
1117 COUNT = 0
1120  HOME : PRINT "  THE PORTIONS OF THE CYCLES BELOW THE  'BAS
ELINE' ARE THE REGENERATIVE PERIODS;THE PORTIONS ABOVE THE 'BASE
LINE' ARE  THE ACTIVE PERIODS (ENERGY STORED IS "
1130  PRINT "AVAILABLE FOR USE)."
1140  HTAB 17
1150  VTAB 8: PRINT "ACTIVE:": VTAB 15: PRINT "——————————————
————————————"
1160  VTAB 22: PRINT "REGENERATIVE:"
1170  VTAB 18: PRINT "(M=MENTAL)": PRINT "(P=PHYSICAL)": PRINT "
(E=EMOTIONAL)"
1180  FOR X = 0 TO 39
1190 A = X / 36 * 2 * 3.1416
1200 AP = A * 36 / 23
1210 AE = A * 36 / 28
1220 AM = A * 36 / 33
1230 YP = 15 -  SIN (AP) * 7 + .5
1240  HTAB X + 1: VTAB YP: PRINT "P"
1250 YE = 15.5 -  SIN (AE) * 7
1260  HTAB X + 1: VTAB YE: PRINT "E"
1270 YM = 15.5 -  SIN (AM) * 7
1280  HTAB X + 1: VTAB YM: PRINT "M"
1290  VTAB 23: NEXT X
1300  REM ********************                              ** "
GRAPHICS ROUTINE FOR  **  SCANNING AND PRINTING  **  OUT THE
CRT. SCREEN.  **                    **********************
*********
1310  PRINT PR$(1); CHR$ (30): REM : PRINTER "ON", AT 40 CHAR/LI
NE.
1315  VTAB 1
1320  FOR SECT = 0 TO 2: FOR LEVEL = 0 TO 7
1330  FOR LOC = 0 TO 39
1340  PRINT  CHR$ ( PEEK (1024 + 40 * SECT + LEVEL * 128 + LOC))
;
1350 COUNT = COUNT + 1: IF COUNT = 200 THEN  PRINT :COUNT = 0: V
TAB  PEEK (37): REM : BY INSERTING A "PRINT" BEFORE 255 CHARACTE
RS ARE PRINTED OUT, AN ODD "SKIP" IN PRINTING IS AVOIDED.
1360  NEXT : NEXT : NEXT : PRINT
1370  REM ********************                              **
END OF SCREEN SCAN.  **  LEAVE PRINTER ON, FOR  **  NEXT LIT
TLE "TAG".  **                    **********************
*********
1410  PRINT "  THE THREE BIO-CYCLES ARE THOUGHT TO  BEGIN TOGET
HER, AT THE MOMENT OF BIRTH." `)
1420  PRINT "THUS, IF AN INDIVIDUAL'S BIRTHDATE IS  KNOWN, THE
PARTICULAR STAGE OF EACH"
1430  PRINT "CYCLE AT ANY GIVEN DATE MAY BE CALCU-  LATED.": PR
INT : PRINT PR$(0)
1435  RETURN
1440  END
24000  DATA (END OF DATA)
25000  REM ********************                              **
HOWIE MITCHELL        **  7823 SW. 55TH PLACE    **  GAINESVI
LLE, FLA. 32601 **  JULY, 1980          **
      ********************************
32599  END
```

A Review of Library Disk #15 and EAMON #1

By Brian Dorner

One of the best, although possibly most overlooked, benefits of belonging to a computer club is the availability of really good programs (at a reasonable price) through the clubs library. Our clubs library, though possibly not as large as some of the more well established clubs, contains a good deal of fine, polished software. The object of the game, then, is to make the club members aware of what is available from the library. That is, alas, a task which of late has been delegated to our illustrious librarian, who is already busy putting together new disks and copying old disks for distibution. It is thus no small wonder that our overworked librarian has requested help in reviewing some of our library disks. Though I am not by any stretch of the imagination a budding author, our librarians plead for assistance in reviewing some of the newer additions to the library have not gone unnoticed, and herein is my attempt to comply with that request.

Rating Scale
------ -----

*****	Superb
****	Better than average
***	Good
**	OK
*	Forget it.

The first disk that we'll take a look at is #15 (GAMES VIII). This disk contains 12 programs, some of you will probably recognize one or two of them, but for the most part these are new programs.

AIR-SEA is a lo-res animated game where the object is to either blast the airplanes or the ships (depending, of course, on whether you are air-force or navy). The more interesting parts of this program are its imaginative graphics (the bombs splash when they hit the water) and sound effects, and the fact that it can be played as a two player game. This program should be on your list if you have kids, or if you happen to be a big kid yourself (aren't we all).
Rating (***)

DEEP-SPACE is the old favorite from Creative Computing. It seems to fare a little weak against some of the newer programs on this disk, but if you're into adventure or space games, this one's a classic and is well worth the price.
Rating (***)

DRAGON Well, if you're into looking at hi-res pictures, don't miss the dragon. I just can't help feeling I've seen him somewhere before (could have been in EAMON).
Rating (**)

FIZZBIN If you're a trekkie I don't have to tell you how to play. If you're not a trekkie, well get this disk and prepare yourself for the wildest, most compilicated card game in the galaxy. A MUST for all loyal Star Trek fans and card sharks.
Rating (****)

GUIDED MISSILE This one's a goodie, as you try to hit the target by guiding the missile with the game paddels - Only it's written in ROM Applesoft and I only have the RAM version. I do have a modified version that works with RAM Applesoft, so if you get this disk and are in the same boat as me, let me know and I'll fixed you up.
Rating (*****)

LASER CANNON It may look easy to hit the flying saucers as they fly over, but, you could end up playing this one for hours. Interesting use of lo-res graphics.
Rating (****)

MADAME DUPRE is a rather sarcastic old fortune teller, she always insults you. Great if you like being insulted by your machine.
Rating (**)

MADLIB is a computerized adaptation of the semi-famous party game. Really good for teaching parts of speech to little ones who hate english.
Rating (***)

ONE PLAYER FOOTBALL Has a somewhat interesting lo-res display, but it wears a little thin with age. If you're a football freak, who knows.
Rating (***)

SAVE THE WORLD One of the 'can't miss' games of the year. Simple enough for even the youngest gamer but, will keep everyone intereted for hours. You control the trajectory of a nuclear bomb with your game paddle and try to destroy 'IT'. IT's great, don't miss IT.

NOTE: There may be a small error in your version, the corrections are below:
line 10 should read:
10 POKE -16302,0: GR : COLOR=7

line 9040 should read:
9040 PB1= PEEK (-16384): IF PB1) 127 THEN 10: GOTO 9040

Rating (****)

SPACE MAZE Here's another one that only runs in ROM Applesoft (I've also got a modified version of this to run on a RAM Applesoft machine). This one looks so easy at first, but just try to pilot your ship through the space maze. It's so easy only a child can do it.
Rating (*****)

Well, that's all for disk #15, now let's take a look at some software from out west.

EAMON is the name of the fantasy role-playing system for the Apple, written by Donald Brown of Role-Playing Starwars fame. This

disk from Denver Apple Pi is definitly one of the best investments of the year. EAMON is an open ended adventure system, meaning EAMON #1 is only the first of (hopefuly) many diffrent adventures (NOTE: EAMON #2 is already available from the library). The adventure on this disk is called The Beginners Cave, and is written specifically to get the adventurer familar with the method of playing Adventure. The disk also contains a number of utilities for maintaining the EAMON system. If you don't have any other game disks, do yourself and your Apple a favor and get EAMON, you won't be sorry.

NOTE: If you have RAM Applesoft you'll have to make a modification to one of the programs. First you should copy the original disk over to a blank disk, then delete all of the utilities and other unused files, put an Applesoft on the disk and make the following modifications.

THE WONDERFUL WORLD OF EAMON
change line 110 to read:
110 HOME :D$ = CHR$(4): PRINT D$;"BLOAD EAMON,PIC,A$4000":
POKE -16304,0:POKE -16297,0:POKE -16302,0:POKE -16299,0

And that's all there is to it...

Well, I hope that the preceding review has been of help to you. Both of these disk are well worth it at twice the price.

Happy Computing !!!

CRAE SOFTWARE REVIEW

by H. S. Pilloff

CRAE is the acronym for CO-RESIDENT APPLESOFT EDITOR which provides global editing capabilities for Applesoft programs. It is available from Highlands Computer Services in Renton, Washington for $19.95 on Disk. CRAE is loaded into memory by running EDIT.LOAD which sets the pointers such that when an Applesoft program is loaded, it locates in memory above CRAE. The editor is activated by entering an ampersand (&) command and a left bracket prompt indicates that CRAE is up. Entering RETURN will deactivate CRAE and restore the familiar Applesoft right bracket prompt.

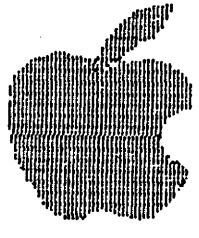
CRAE use single letter commands:
A --- Append program on disk to one in memory
C --- Change any string in program
D --- Formatted hex dump
F --- Find any string in program
L --- Optimized list of program
N --- Auto line numbering
Q --- Quote a range of line numbers from one area to another
R --- Renumber

On receipt of this software I was surprised to find a stamped, self-addressed user evaluation form. I volunteered some comments and subsequently received a second disk with a new version CRAE 2 together with a four page letter. (They sent the letter because they had been unable to reach me by phone. All this for $15! The price has since increased.) Because in my opinion CRAE 2 is so much superior to CRAE 1, these comments apply to CRAE 2 even though this version is not expected to be available for several months. In addition to all the commands in CRAE 1 a number of new and very useful commands have been added. In particular 'M' is a line modifying command which provides for line editing capabilites including insert/delete as well as I,M,J,K cursor control. Other new commands include:

* --- precedes monitor commands
! --- prints number of free bytes
!XXXX --- decimal equivalents of hex 0-FFFF
% --- decimal length and address of last BLOAD program
%XXXXX --- hex equivalent of decimal 0-65535

I tend to write very long programs (a necessity - definitely not a virtue) and use extensively two utilities: Konzen's PLE and CRAE 2. They are to a considerable extent mutually complimentary (line oriented vs. global) in their features and I often go from one to the other. (The ideal utility would include the features of both.) This poses some difficulties as the two routines are to some extent mutually exclusive. With the PLE up, it is not altogether surprising that I have not been able to load CRAE. As a result, it is necessary to BRUN REMOVE PLE (available on PLE ver. 2) in order to run EDIT.LOAD and then activate CRAE.

In summarizing my own experiences I have found CRAE to be an extremely useful utility. The documentation is good and the continued support of the vendor to improving this product have combined to produce a high quality offering.



Washington Apple Pi
Membership Application

Dues for membership in Washington Apple Pi are $12.00 per year. The dues year runs from January 1 to December 31. Members joining after February should pay at the rate of $1.00 per month. Both new members and renewing members are asked to fill in the following application as completely as possible. Information gained here will help the club serve you better.

If you have any questions please call Genevie Urban (301 229-3458) or Bob Peck, Treasurer (301 468-2305).

Thank you.

NAME_____

ADDRESS_____

CITY, STATE, ZIP _____

TELEPHONE NUMBERS: HOME ( )_____ WORK ( )_____

PLEASE LIST HARDWARE YOU OWN: APPLE II__ APPLE II PLUS__ 48K__32K__24K__16K__
DISK II__ SERIAL__/PARALLEL__I/O AI/O__RS-232__

OTHER:(e.g. MODEM, PRINTER)

_____

_____

OCCUPATION_____

I WOULD LIKE TO WRITE ARTICLES FOR THE NEWSLETTER (Y/N)_____

MY AREAS OF INTEREST ARE:_____

_____

IF YOU WOULD LIKE TO ASSIST ON A COMMITTEE OR SPECIAL INTEREST GROUP, PLEASE SPECIFY_____

_____

I AUTHORIZE THE RELEASE OF MY NAME__, ADDRESS__ AND TELEPHONE__ TO OTHER MEMBERS.

(NOTE: Club policy prohibits releasing members' names and addresses unless you release that information by checking the appropriate areas above.)

PLEASE ENCLOSE PAYMENT WITH THIS APPLICATION.

MAKE CHECK PAYABLE TO "WASHINGTON APPLE PI"

MAIL TO: WASHINGTON APPLE PI Attn. Treasurer
PO BOX 34511
WASHINGTON, DC 20034

WASHINGTON APPLE PI
MAIL ORDER FORM

Washington Apple Pi now has a program library, and disks are available for purchase by anyone. The price to members is $5.00 per disk, and $8.00 to non-members. These disks are chock full of exceptional programs - the utilities are especially useful. The games are some of the best - not just simple and uninteresting ones. You may pick them up at any meeting or have them mailed for $2.00 per disk additional. They will come in a protective foam diskette mailer.

Also available for purchase by members at a discount price is the new APPLE II REFERENCE MANUAL (replaces the Red Reference Manual). The price of this manual is $17.00. You may pick it up at a meeting or have it mailed to you at no extra charge.

Amount

1. New APPLE II REFERENCE MANUAL - $17.00 each ------

2. PROGRAM DISKETTES
   Members: $5.00 per disk picked up at meeting
   $7.00 mailed to you...
   Non-members: $8.00 per disk picked up a meeting
   $10.00 mailed to you...

Volume 1--Utilities I	()	Volume 181--Eamon #1	()
Volume 2--Utilities II	()	Volume 182--Eamon #2	()
Volume 3--Games I	()	(Both Disks Required	
Volume 4--Games II	()	To Play Game)	
Volume 5--Games III	()		
Volume 6--Games IV	()		
Volume 7--Games V	()		
Volume 8--Utilities III	()		
Volume 9--Educational I	()		
Volume 10-Math/Science	()		
Volume 11-Graphics I	()		
Volume 12-Games VI	()		
Volume 13-Games VII	()		
Volume 14-IAC Utilities IV	()		
Volume 15-Games VIII	()		
Volume 16-Utilities V	()		
Volume 17-Graphics II	()		
Volume 18-Education II	()		
Volume 19-Communications	()		
Volume 20-Music	()		
Volume 21-Apple Orchard	()		

TOTAL ORDER = $ ------

Check here if you want these shipped---

NAME ----------

ADDRESS ----------

CITY, STATE, ZIP ----------

TELEPHONE ----------

Membership No.(1st three digits after WAP on mailing label) ---------

Make checks payable to "Washington Apple Pi"

Send order to: Washington Apple Pi- ATTN: Librarian
PO Box 34511
Washington, DC 20034