TI-D-BITS

PHILADELPHIA AREA USERS GROUP NEWSLETTER
COVERING THE TI99/4A
AND MYARC 9640 COMPUTERS

MAY 1990

Volume 10 Number 5

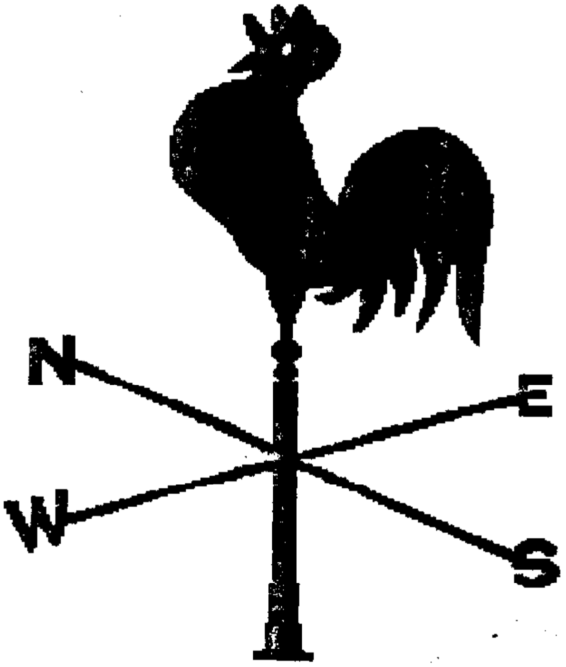
THERE ARE STILL MANY DIRECTIONS TO LOOK FOR TI AND GENEVE SUPPORT

FAIRWARE PROGRAMMERS

HARDWARE DEALERS

COMMERCIAL SOFTWARE DEALERS

USER GROUPS



TI CARES

BBS'S

THE PHILADELPHIA AREA TI-99/4A USERS' GROUP (MAY '90)

The Philadelphia Area TI-99/4A Users' Group meets twice a month. On the first Saturday of any given month, we meet at the Bucks County Youth Development Center, (YDC, which is next to Neshaminy Mall), Administration Building, beginning at 10:00 am. On the third Saturday of each month, we meet at LaSalle University, 20th Olney, in room H-327 located in the Science Building. Membership to The Philadelphia Area TI-99/4A Users' Group is available to all. We invite anyone that is interested in the TI-99/4A to visit us. Stop in and see what is available to you for your TI and how membership can benefit you!

Current executive board consists of:

PRESIDENT...........	Allan Silversteen.	215-885-7910
VICE PRESIDENT......	Eric Bray.........	215-848-5515
SECRETARY...........	Mark Wannop.......	609-365-1776
TREASURER...........	Don Arsenault.....	215-368-0446

Committees consists of:

TI-d-BITS	Ralph Field.......	215-362-2534
	Don Arsenault.....	215-368-0446
	OPEN	
	Rice Hall	
LIBRARY	Ted Chemey........	215-752-1458
	Rich Mascara	215-441-4060
MEMBERSHIP ...	OPEN	
ASSISTANT TREASURER.	Frank Passini	
EDUCATION	Barry Traver	
	Frank Passini	
	Ted Chemey	
	Tim Coyne	
	Carlo Angelico	
EQUIPMENT	Rice Hall	
PROGRAM	Dr. Eric Bray	

REMEMBER to be considerate when calling any of the above people. Limit your calls to the early evening hours. (6pm to 9pm)

The opinions expressed herein are those of the individual authors are not necessarily those of the Philadelphia Area TI-99/4A Users' Group or its officers. Nor is the Philadelphia Area TI-99/4A Users' Group or any of its officers responsible for any damage, inconvenience, or loss which may result as a consequence of the use of any written material herein.

TI-d-Bits is published monthly by the Philadelphia Area TI-99/4A Users' Group, c/o Don Arsenault, 1290 Buttonwood Dr., Lansdale, PA 19446. All material herein may be reprinted freely by other non-profit User Groups, (unless otherwise stated), as long as proper credit is given to both source and author. Contributions are encouraged, but no payment is made. Editorial, advertising, and classified, copy MUST be in by the LAST day of the previous month. You can either mail your copy to: TI-d-Bits, c/o The Philadelphia Area TI-99/4A Users' Group, c/o Don Arsenault, 1290 Buttonwood Dr., Lansdale, PA 19446 or send it via modem by contacting Don Arsenault at (215)-368-0446. If your piece contains any diagrams, charts, or code, send a paper copy AT FINAL PUBLICATION SIZE.

Classified ads are printed in blocks. A block consists of 3 lines, 55 characters wide, or any increment of 3 lines. Classified advertising is accepted from members at NO CHARGE for a one block ad, per issue. Additional ads from members may be placed at cost of $1.00 per block. Non-members may place classified ads at a cost of $2.00 per block. All advertisements MUST be paid for in advance.

Commercial advertising is accepted for publication at the following rates:

Quarter page	$ 5.00
Half page	$ 8.00
Full page	$15.00

Commercial advertisements will be placed in the next available issue. All advertisements MUST be paid for in advance.

The editor of TI-d-Bits or the executive board of The Philadelphia area TI-99/4a Users' Group reserve the right to reject any material submitted for publication for any reasons.

The Philadelphia Area TI-99/4A Users' Group's program library is available to all active members at NO CHARGE for copying to your disk. A charge of $2.00 per disk is made for club supplied disks for members. Non members may obtain copies of the library for a fee of $5.00 per disk. A catalog of the library's contents is given to all new members upon request and updates will appear in this publication from time to time. To obtain material from the library, contact the librarian for the best procedure to obtain your requests.

########################

NEW-AGE/99

By Jack Sughrue
Box 459
East Douglas, MA 01516

########################

I often chuckle at doomsayers, but sometimes they irk me.

I don't think anyone questions the fact that the APPLE, AMIGA, IBM, and clones all have more commercial software and hardware support than the 4A. Walk into any bookstore and look at the magazines. Go into a department store and look at the racks of software packages. Or into an electronics store and peruse the computers, cards, drives, other hardware and software items. And the prices.

No question: the stuff's there. And some of us succumb to those temptations. Don't get me wrong. At work and at the homes of friends and relatives I get plenty of opportunities to play and work with these other machines. And enjoy my time on them, for the most part. But, even then and even when I attend the other computer user groups or fairs, I don't have the feeling that I do with similar TI activities.

The emotion - very much in evidence at TI get-togethers - is absent at Apple and IBM gatherings. There are subtle, important other differences, too.

When I look at the Other software or Other hardware, I wonder if most TIers would pay those humongous prices or whether most TIers would want the trouble to learn these new ways of hi-tech wizardry. For the most part, the TIers don't do that now.

An example: The Apple IIGS is $1149. (That's without the 5.0 (NOTE THE CONSTANT UPGRADING THAT SEEMS TO BE A COMPLAINT IN SOME CIRCLES!) $50 Systems disk that's required to operate the machine. Nothing resides in memory.)

Let's say you want a word processor for it, as word processing is the most common use of non-business computers. Appleworks (the most popular WP for all Apples) is another $250. (no percentage point, folks. It's two-hundred-fifty dollars, plus tax!) Will the WORKS give you 40/80 column for those of us (oldtimers?) who LIKE 40 because the letters are large and clean and easy to read? Nope. Try reading 80 on an Apple monitor.

Can you flip around, as you can with the fairware FUNNELWEB, for example, and load up such utilities as DM-1000, Disk Utilities, ARChiver, other languages (Assembly, c, FORTH), other sources(tape, cartridge, etc.)? Nope. There are no tapes and cartridges for those Other machines, anyway, but the WORKS doesn't let you configure any possibilities outside the provided environments. How about modified fonts, underlining, doublestrike? All available with FUNNELWEB, even more so if you use the fairware PLUS! within the FWB environment.

And speed? The IIGS is slow, very slow. But for an additional $399 you can buy a TRANSWARP GS card that'll speed up WORKS and other GS items to reasonable, runnable speed.

Can you slip into graphics with WORKS? Nope, but FWB's TI text can easily emigrate to PAGEPRO for all kinds of graphics/text manipulations (or PP can stand alone for similar structuring).

You must use PRINT SHOP separately to get some graphics; still, not with the page possibilities. For that you'd need the user UNfriendly NEWSROOM. Add another $400. Not counting the graphics. The kind that are Public Domain through TIPS for the TI. Say another $500, easily. How much is that decidedly inferior word processor now? Still under $3000? That's not bad as prices go in the computer world out there. Check Other computer Other prices.

Why would we TIers consider THAT an upgrade? We certainly wouldn't pay those prices for software for our superior machine; FWB donation $20; PLUS! donation $10; TIPS and 5,000 graphics are FREE; PAGEPRO is under $25. Grand total for us TIers, maybe 50 bucks at the most.

Most TIers don't have RAM disks or RAVE keyboards or hard drives or the GENEVE upgrade that includes 640K RAM, truly outstanding graphic capabilities, a superb keyboard and all kinds of other great stuff, including TI compatibility (as much as most clones have with IBM). And yet EACH of these things are less than a couple pieces of software for Other computers.

Most TIers don't (in all honesty) even pay for the fairware they use, so I can't see them spending $50 to $800 per piece of software after spending a couple thousand for another system, no matter how great the software is. Look at the magnificent under $25 commercial software items for the TI: TI_BASE, PAGEPRO, TI ARTIST PLUS, for example. Do most TIers who have disk drives own these three items? Not by a long shot!

Have most of the TIers who use Tony McGovern's FUNNELWEB, Barry Boone's ARCHIVER, John Birdwell's DISK UTILITIES, or Canada's DM_1000 sent decent (or any) fairware contributions to the authors? Or contributions for the constant updates? NOPE.

Do most TIers subscribe to **MICROpendium** or **ASGARD NEWS**, the only two magazines we have devoted entirely to the TI? **NOPE.**

Do most TIers even belong to user groups? **NOPE.** Not even by mail, though that is the best source of disk, tape, and text materials, not to mention the monthly newsletter connection, that money can buy (also under $25).

Do most TIers take advantage of the massive sources available across their phone through inexpensive modems? **NOPE.**

My feeling is why, if TIers are not even taking advantage of all the things that are available <u>NOW</u> and at a lot smaller cost, would they even want to move (up?) to more expensive machines?

It makes me think about a comment by Keith Jarrett, considered by many music critics to be the greatest pianist of this century. Because he plays jazz and classical and newage and a style of improv that is inimitable, he was asked why he didn't play the electronic keyboard. "I haven't learned everything about the acoustic piano, yet.", he said. If he hasn't learned everything about the acoustic piano yet, no one in the world has.

But I feel this way about my TI. When people ask me why I don't move up, I first give them my speech about moving to Apples or clones or whatever is not necessarily UP. Then I think of the real reason: I haven't finished learning everything I want to learn and doing everything I want to do with my 4A yet.

Even if everything relating to TI ceased instantly; nothing more being created; all user groups stopped; the complete TI end - even if... most of us would still be using and learning and enjoying our perfectly wonderful computer for a long time to come.

But such a scenario is not heading our way in the near future. There are too many of us who care and **WANT** to stay with this gem of a machine.

I think we all have a long way to go yet, and I am enjoying the journey.

><$><$><$><$><$><$><$><$><

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Playing With the Line Number Table

By Ken Woodcock
Tidewater TI-99 User's Group

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Actually we're not going to be manipulating the LNT, just using it to play some tricks with Extended Basic programs. In my past articles, **ANALYZING A BASIC PROGRAM**, I explained how the LNT is used when listing a program. Now let's see how we can make a program "**UNLISTABLE**"! The trick is to modify the first byte of the line data. The first byte indicates how long the line is. All we have to do to make it unlistable is to change that byte to zero.

Here is a short XB program that will do it. The first step is to type in the program and save it in **MERGE** format (**SAVE DSK1.FIX,MERGE**). Now load any Basic/XB program, RESequence it if necessary so that no lines are lower than 10, then merge in FIX (**MERGE DSK1.FIX**), and RUN it. The program shows each line number as it sets it's length byte to zero. When it finishes, try to list it! Yeh, just try!! A pretty screen isn't it? If you don't have a color monitor you won't appreciate it as much. Anyway, your computer will lock up and will have to be reset. This does not prevent the program from being copied or resaved but does inhibit modifications. Some commercial software companies and even some "freeware authors" have used this technique to "PROTECT" their software.

FIXing a program to be unlistable by using the above method was the easy part. The length byte simply is set to zero regardless of what it was; reversing this procedure is a bit more tricky. I have "UNFIXED" some programs manually by getting a sector printout with **DISKFIXER** or **DISK UTILITIES**, then scanning the list for 2 consecutive zero bytes (00 00). While this worked, it was both tedious and prone to error; two conditions that are just begging for a computer solution. So I set out to write a program to do the work. My first attempt was to put >FF in all the length bytes. This was simple to do and did allow the program to be listed but had some undesirable side effects when editing lines.

The length byte had to be accurately determined. Since the end of each line is marked with a zero, I tried to count the bytes from the start of the line to a zero but a zero byte can appear within a line when a line number is referenced in a GOTO, GOSUB, etc.. In the **UNFIX** program the method used is to count bytes until a zero is found. Then the LNT is checked to see if this is a true end of line by comparing its address to the start of another line and to the end of the program. This process is somewhat slow but much faster and more accurate than doing it manually. It is used in the same manner as the **FIX** program: first load the unlistable program, RESequence it, **MERGE DSK1.UNFIX**, **RUN**. Now you can delete lines 1-6 and save your recovered program.

(save as FIX in MERGE form)

```
1 CALL INIT :: CALL PEEK(-31952,A,B,C,D):: SL=C*256+D-6553
9 :: EL=A*256+B-65536 :: FOR X=SL TO EL STEP -4 :: CALL PE
EK(X,E,F,G,H):: ADD=G*256+H-65536 :: PRINT "LINE # ";E*256
+F
2 CALL LOAD(ADD-1,0):: NEXT X ::STOP :: !@P-
```

(save as UNFIX in MERGE format)

```
1 CALL INIT :: CALL PEEK(-31952,A,B,C,D):: SL=C*256+D-6553
9 :: FOR X=SL TO EL STEP -4 :: CALL PEEK(X,E,F,G,H):: ADD
=G*256+H-65536 :: PRINT "LINE # ";E*256+F
2 I=1 :: CALL PEEK(ADD-1,V):: IF V THEN 6
3 CALL PEEK(ADD+I,V,W):: IF V THEN I=I+1 :: GOTO 3
4 FOR Y=SL TO EL STEP -4 :: CALL PEEK(Y,E,E,E,F):: IF
 E*256+F-65536=ADD+I+2 OR W=0 OR ADD+I)-3 THEN CALL LOAD(A
DD-1,I+1):: GOTO 6
5 NEXT Y :: I=I+1 :: GOTO 3
6 NEXT X :: STOP :: !@P-
```

)(*)(*)(*)(*)(*)(*)(*)(*)(

TIPS FROM THE TIGERCUB

#54

Copyright 1988

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

Distributed by Tigercub Software to TI-99/4A Users Groups for promotional purposes and in exchange for their newsletters. May be reprinted by non-profit users groups, with credit to Tigercub Software.

Over 120 original programs in Basic and Extended Basic, available on cassette or disk, NOW REDUCED TO JUST $1.00 EACH!, plus $1.50 per order for cassette or disk and PP&M. Minimum order of $10.00. Cassette programs will not be available after my present stock of blanks is exhausted. The Handy Dandy series, and Color Programming Tutor, are no longer available on cassette.
Discriptive catalogs, while they last, $1.00 which is deductable from your first order.

Tigercub Full Disk Collections, reduced to $5 postpaid. Each of these containes either 5 or 6 of my regular catalog programs, and the remaining disk space has been filled with some of the best public domain programs of the same category. I am NOT selling public domain programs - they are a free bonus!

TIGERCUB'S BEST, PROGRAMMING TUTOR, PROGRAMMER'S UTILITIES, BRAIN GAMES, BRAIN TEASERS, BRAIN BUSTERS!, MANEUVERING GAMES, ACTION GAMES, REFLEX AND CONCENTRATION, TWO-PLAYER GAMES, KID GAMES, MORE GAMES, WORD GAMES, ELEMENTARY MATH, MIDDLE/HIGH SCHOOL MATH, VOCABULARY AND READING, MUSICAL EDUCATION, KALEIDOSCOPES AND DISPLAYS.

NUTS & BOLTS DISKS These are full disks of 100 or more utility subprograms in MERGE format, which you can merge into your own programs and use, almost like having another hundred CALLs available in Extended Basic. Each is accompanied by printed documentation giving an example of the use of each.
NUTS BOLTS (No.1) has 100 subprograms, a tutorial on using them, and 5 pp. documentation. NUTS & BOLTS No.2 has 108 subprograms, 10 pp of documentation. NUTS & BOLTS #3 has 140 subprograms and 11 pp. of documentation. NOW JUST $15 EACH, POSTPAID.

TIPS FROM THE TIGERCUB
These are fill disks which contain the programs and routines from the Tips from the Tigercub newsletters, in ready-to-run program format, plus text files of tips and instructions.
TIPS (Vol.1) contain 50 original programs and files from Tips newsletters No.1 through No.14. TIPS VOL.2 containes over 60 programs and files from Nos.15 thru 24. TIPS VOL.3 has another 62 from Nos.25 through 32. TIPS VOL.4 has 48 more from issues No.33 through 41.
NOW JUST $10 EACH POSTPAID.

```
****************************
*         NOW READY        *
* TIPS FROM TIGERCUB VOL.5 *
* Another 49 programs and  *
* files from issues No.42  *
* through 50. Also $10 ppd *
****************************
```

TIGERCUB CARE DISKS #1,#2,#3 AND #4. Full disks of text files (printer required). No.1 containes the Tips news letters #42 thru #45, etc. Nos.2 and 3 have articles mostly on Extended Basic programming. No.4 containes Tips newsletters Nos. 46-52. These were prepared for user groups newsletter editors but are available to anyone else for $5 each postpaid.

This program uses the program that writes a program technique to create a program that can be used over and over to create quiz programs.
When you key in these routines, DON'T change any line numbers!

First key in this routine and run it to create a D/V 163 file names ASCII on the disk in drive 1.

```
100 OPEN #1:"DSK1.ASCII",VARIABLE 163,OUTPUT
110 FOR J=1 TO 125 :: X$=X$&CHR$(J):: X2$=X2$&CHR$(J+125
):: NEXT J
120 PRINT #1:CHR$(0)&CHR$(230)&"X$"&CHR$(190)&CHR$(199
)&CHR$(125)&X$&CHR$(0)
13 PRINT #1:CHR$(0)&CHR$(240)&"X2$"&CHR$(190)&CHR$(199
)&CHR$(125)&X2$&CHR$(130)&"J$"&CHR$(190)&"X$"&CHR$(
184)&"X2$"&CHR$(0)
140 PRINT #1:CHR$(255)&CHR$(255)
```

Next, key in this part -

```
220 CALL CLEAR :: CALL SCREEN(5):: FOR SET=1 TO 12 ::
CALL COLOR(SET,2,B):: NEXT SET :: DIM L$(250,4)
230 !skip to line 280!
280 READ M$ :: DISPLAY AT(2,14-LEN(M$)/2):M$ :: FOR J-1
TO C :: READ M$ :: DISPLAY AT(6+J,4):J;M$ :: NEXT J
290 DISPLAY AT(12,1):"Category to match?
(1-"&STR$(C)&")" :: ACCEPT
AT(12,26)SIZE(1)VALIDATE("1234"):M :: IF M>C THEN 290
300 IF C=2 AND M=1 THEN A=2 :: GOTO 320 ELSE IF C=2 AND
M=2 THEN A=1 :: GOTO 320
310 DISPLAY AT(14,1):"Match against (1-"&STR$(C)&")" ::
ACCEPT AT(14,21)SIZE(1)VALIDATE("1234"):A :: IF A>C OR
A=M THEN 310
320 DISPLAY AT(16,1):"How many choices? (2-5)" :: ACCEPT
AT(16,25)SIZE(1)VALIDATE("2345"):CH :: IF CH>N-1 THEN 320
330 FOR J=1 TO N :: FOR L=1 TO C :: READ L$(J,L):: NEXT L
:: NEXT J
340 X$=SEG$(J$,1,N):: FOR J=1 TO CH :: RANDOMIZE ::
X=INT(LEN(X$)*RND+1):: Y(J)=ASC(SEG$(X$,X,1))::
X$=SEG$(X$,1,X-1)&SEG$(X$,X+1,255):: NEXT J
350 Z=INT(CH*RND+1):: IF L$(Y(Z),1)=Y$ THEN 350 ELSE
Y$=L$(Y(Z),1)
360 DISPLAY AT(8,1)ERASE ALL:L$(Y(Z),M):: FOR J=1 TO CH
:: DISPLAY AT(10+J,4):J;L$(Y(J),A)
370 NEXT J :: DISPLAY AT(23,1):""
380 DISPLAY AT(20,1):"(1-";STR$(CH);")?" :: ACCEPT
AT(20,8)SIZE(1)VALIDATE(DIGIT):Q :: IF Q-0 OR Q>CH THEN
380
390 IF L$(Q,M)<>L$(Z,M)THEN 410 :: DISPLAY
AT(23,1):"CORRECT!"
400 CALL SOUND(100,659,5):: CALL SOUND(100,784,5):: CALL
SOUND(400,1047,5):: GOTO 340
410 DISPLAY AT(23,1):"WRONG!" :: CALL
SOUND(300,110,0,-4,5):: GOTO 380
```

Enter MERGE DSK1.ASCII and then SAVE DSK1.QUIZ,MERGE Then key in -

```
100 OPEN #1:"DSK1,QUIZ",VARIABLE 163,INPUT :: OPEN
#2:"DSK1.QUIZ/2",VARIABLE 163,OUTPUT
110 FOR J=220 TO 410 STEP 10 :: LINPUT #1:M$ :: CALL
LINE(J,LN$)
120 PRINT #2:LN$&CHR$(156)&CHR$(253)&CHR$(200)&CHR$(1)
&"1"&CHR$(181)&CHR$(199)&CHR$(LEN(M$))&M$&CHR$(0):: NEXT J
130 PRINT #2:CHR$(255)&CHR$(255):: CLOSE #1 :: CLOSE #2
140 SUB LINE(LN,LN$):: LN$=CHR$(INT(LN/256))&CHR$(LN-256*
INT(LN/256)):: SUBEND
```

Run that to convert the merge file QUIZ into another merge file QUIZ/2. Then key this in -

```
100 CALL CLEAR :: CALL SCREEN(5):: FOR SET=1 TO 12 ::
CALL COLOR(SET,2,8):: NEXT SET :: DISPLAY
AT(2,5):"TIGERCUB QUIZWRITER"
110 CALL CHAR(64,"3C4299A1A199423C"):: DISPLAY AT(4,1):"
Tigercub Software for free":"distribution - no copying":
"fee may be charged."
120 DISPLAY AT(8,1):"This program will write":"multiple
choice quizzes of":"the category match type."
130 DISPLAY AT(11,1):"It will accept up to 250":"records,
if memory permits,":"and up to 4 categories
per":"record."
140 DISPLAY AT(15,1):"For instance, a quiz on the":"table
of elements could have":"the elements name, its
symbol":"and its atomic weight."
150 DISPLAY AT(19,1):"The program will allow you":"to
select which two cate-":"gories to match
160 DISPLAY AT(23,8):"PRESS ANY KEY" :: DISPLAY
AT(23,8):"press any key" :: CALL KEY(5,K,S):: IF S=0 THEN
160 ELSE CALL CLEAR
170 DISPLAY AT(2,1):"The Quizwriter can be used":"over and
over to write any":"number of different quizzes,"
180 DISPLAY AT(5,1):"and each quiz can be SAVEd":"and run
again and again."
190 DISPLAY AT(12,1):"Place a disk in drive 1
with":"enough space available for":"the quiz."
200 DISPLAY AT(15,1):"What filename will you use":"for
the quiz?":"DSK1." :: ACCEPT AT(17,6):F$ :: CALL CLEAR
210 OPEN #1:"DSK1."&F$,VARIABLE 163,OUTPUT
220 !SKIP TO LINE 420!
420 DISPLAY AT(8,1):"TITLE OF QUIZ?" :: ACCEPT AT(10,1):T$
430 T$=CHR$(147)&CHR$(200)&CHR$(LEN(T$))&T$ :: DISPLAY
AT(12,1):"NUMBER OF CATEGORIES (2-4)?"
440 ACCEPT AT(12,28)SIZE(1)VALIDATE("234"):C :: PRINT
#1:CHR$(0)&CHR$(250)&"C"&CHR$(190)&CHR$(200)&CHR$(1)
&STR$(C)&CHR$(0)
450 FOR J=1 TO C :: DISPLAY AT(12+J*2,1):"CATEGORY
#";STR$(J);" TITLE?" :: ACCEPT AT(13+J*2,1):C$(J)
460 T$=T$&CHR$(179)&CHR$(200)&CHR$(LEN(C$(J)))&C$(J)::
NEXT J
470 PRINT #1:CHR$(1)&CHR$(14)&T$&CHR$(0)
480 DISPLAY AT(2,1)ERASE ALL:"INPUT DATA":;:"(input END
when finished)"
490 N=N+1 :: Z$="" :: DISPLAY AT(6,1):"RECORD
#"&STR$(N)&RPT$("          ",200):: FOR J=1 TO C :: DISPLAY
AT(7+J,1):C$(J):: ACCEPT AT(8+J,1)SIZE(20):Y$
500 IF Y$-"END" THEN N-N-1 :: GOTO 530
510 Z$=Z$&CHR$(200)&CHR$(LEN(Y$))&Y$&CHR$(179):: NEXT J
```

```
520 LN=1000+N*10 :: CALL LINE(LN,LN$):: PRINT
#1:LN$&CHR$(147)&SEG$(Z$,1,LEN(Z$)-1)&CHR$(0):: GOTO 490
530 PRINT #1:CHR$(1)&CHR$(4)&"N"&CHR$(190)&CHR$(200)&
CHR$(LEN(STR$(N)))&STR$(N)&CHR$(0)
540 PRINT #1:CHR$(255)&CHR$(255):: CLOSE #1
550 DISPLAY AT(8,1)ERASE ALL:"Enter NEW":;:"Enter MERGE
DSK1."&F$:;:"Enter SAVE DSK1."&F$:;:"RUN" :: END
560 SUB LINE(LN,LN$):: LN$=CHR$(INT(LN/256))&CHR$(LN-256*
INT(LN/256)):: SUBEND
```

Enter MERGE DSK1.QUIZ/2 and SAVE the result as your completed QUIZWRITER.

This truly remarkable one-line disk cataloger tinygram by John Martin was published in the Jackson County newsletter -

```
1 IF F THEN INPUT #1:A$,A,J,K :: IF J THEN PRINT A$;TAB(
12);J;TAB(18);SEG$(B$,ABS(A*2)+1,2);K;TAB(27);A<0 :: GOTO
 1 ELSE RUN ELSE B$="AVDFDVIFIVPG" :: INPUT "DSK":F ::
OPEN      #1:"DSK"&STR$(F)&".",INTERNAL,RELATIVE,INPUT ::
GOTO 1 ! BY JOHN M
```

And an ingenious tinygram version of Wheel of Fortune, in the West Penn newsletter.

```
1 ! *** FORTUNE OF WHEELS **
   *      A TINYGRAM       *
   *by Mike &  Ed Machonis*
   ************************
2 CALL CLEAR :: INPUT "ENTER THE MYSTERY PHRASE    "M$
:: CALL CLEAR :: L=LEN(M$)
3 D$-RPT$(CHR$(30),L) :: FOR J=1 TO L :: IF
SEG$(M$,J,1)<>" " THEN 4 ELSE D$=SEG$(D$,1,J-1)&"
"&SEG$(D$,J+1,L)
4 NEXT J :: PRINT D$
5 T=T+1 :: PRINT :"TRY No.";T:;: :: INPUT "TYPE LETTER OR
ENTIRE PHRASE":A$ :: IF LEN(A$)>1 AND LEN(A$)<L THEN 5
6 W=L+1-T :: IF A$=M$ THEN 9
7 FOR J=1 TO L :: IF SEG$(M$,J,1)=A$ THEN
D$=SEG$(D$,1,J-1)&A$&SEG$(D$,J+1,L)ELSE 8
8 NEXT J :: PRINT :D$ :: GOTO 5
9 FOR J=1 TO M :: CALL SOUND(200+J*10,330+40*J,0):: NE
J :: PRINT :"YOU WIN ";STR$(W);",000 WHEELS!":;: :: INPUT
"PRESS ENTER TO PLAY AGAIN":G$ :: T=0 :: GOTO 2
```

```
100 ON WARNING NEXT :: DISPLAY AT(3,10)ERASE
ALL:"KALKULATOR":"":"Input 1st value and Enter.":"Input
other values preceded":"by +,-,* or / and Enter." ! by
Jim Peterson
101 DISPLAY AT(8,1):"Input = and Enter to get":"final
result."
110 R=14 :: C=1 :: ACCEPT AT(12,1):N :: V-N :: F-1 ::
N$=STR$(N):: GOSUB 200
120 ACCEPT AT(12,1)VALIDATE("+-*/=",NUMERIC):N$ ::
A=POS("+-*/=",SEG$(N$,1,1),1):: GOSUB 200 :: IF A=0 THEN
120 :: IF A=5 THEN 160
130 ON ERROR 140 :: N=VAL(SEG$(N$,2,LEN(N$)-1)):: GOTO 150
140 CALL SOUND(200,110,5,-4,5):: C=C-LEN(N$):: DISPLAY
AT(R,C):"" :: RETURN 120
150 IF A=1 THEN V=V+N :: GOTO 120 ELSE IF A=2 THEN V=V-N
:: GOTO 120 ELSE IF A=3 THEN V=V*N :: GOTO 120 ELSE IF
A=4 THEN V=V/N :: GOTO 120
160 DISPLAY AT(R,C);STR$(V):: F,V=0 :: GOTO 110
200 DISPLAY AT(R,C):N$ :: C=C+LEN(N$):: IF C>20 THEN C=1
:: R=R+1 :: RETURN ELSE RETURN
```

Here is the world's shortest tic-tac-toe game, by R.Walters, converted to a tiny gram by Jim Peterson

```
2 DISPLAY AT(5,1)ERASE ALL:"LET'S PLAY TIC-TAC-TOE"
::TAB(10);"1 2 3": :"TAB(10);"8 9 4": :TAB(10);"7 6 5":
3 A=9 :: GOSUB 8 :: S=B
4 DEF F(X)=X-4+4*SGN(8.5-X)
5 C=F(S+1):: GOSUB 6 :: C=F(S+3):: GOSUB 6 :: C=F(S+6)::
IF S/2=INT(S/2)THEN 7 :: DISPLAY AT(20,1):"I MOVE
TO";F(S+4):"":"THE GAME IS A DRAW" :: STOP
6 A=C :: GOSUB 8 :: H=B :: IF H<>F(C+4)THEN 7 ELSE RETURN
7 DISPLAY AT(20,1):"I MOVE TO";F(C+4);"AND WIN!" :: END
8 DISPLAY AT(20,1):"I MOVE TO":A:"":"WHERE DO YOU MOVE
TO?" :: ACCEPT AT(22,23)VALIDATE("12345678"):B :: RETURN
```

```
1 ! STRIGHT-LINE CALCULATOR
    TINYGRAM by Jim Peterson
    Accepts input such as
    6+6-11*2+3/4
2 T,F=0 :: C$="+-*/" :: ACCEPT AT(12,1)ERASE ALL
VALIDATE(NUMERIC,C$):F$ :: L=LEN(F$):: FOR J=1 TO L ::
X$=SEG$(F$,J,1):: P=POS(C$,X$,1):: IF P=0 THEN 5
3 IF F=0 THEN T=VAL(SEG$(F$,1,J-1)):: F=1 :: A=J+1 ::
P2=P :: GOTO 50
4 V=VAL(SEG$(F$,A,J-A)):: A=J+1 :: GOSUB 7 :: P2=P
5 NEXT J :: V=VAL(SEG$(F$,A,255)):: GOSUB 7 :: DISPLAY
AT(12,L+1):"=";STR$(T)
6 DISPLAY AT(24,1):"PRESS ANY KEY" :: CALL KEY(0,K,S)::
IF S=0 THEN 6 ELSE 2
7 IF P2=1 THEN T=T+V ELSE IF P2=2 THEN T=T-V ELSE IF P2=3
THEN T=T*V ELSE T=T/V
8 RETURN
```

That's all, folks!

```
100 DISPLAY AT(3,10)ERASE AL
L:"GO SEARCH": :TAB(7):"by J
im Peterson"
110 DISPLAY AT(7,1):" For us
e before modifying a":"progr
am.":" Searches a program sa
ved in":"MERGE format, finds
 all":"lines containing a ju
mp."
120 DISPLAY AT(12,1):"sorts
into `to' line number":"sequ
ence, outputs to screen":"or
 printer."
130 DATA GOTO,GOSUB,THEN,ELS
E,RUN,RETURN,RESTORE,USING,E
RROR
140 DIM C(200):: FOR J=1 TO
9 :: READ GO$(J):: NEXT J ::
 A=1
150 DISPLAY AT(16,1):"FILENA
ME? DSK" :: ACCEPT AT(16,14)
:F$
160 ON ERROR 170 :: OPEN #1:
"DSK"&F$,INPUT ,VARIABLE 16
 :: GOTO 180
170 DISPLAY AT(18,1):"I/O ER
ROR" :: ON ERROR STOP :: RET
URN 150
180 DISPLAY AT(18,1):"OUTPUT
 TO P":"  (S)creen":"  (P)ri
nter" :: ACCEPT AT(18,11)SIZ
E(-1)VALIDATE("PS"):Q$
190 IF Q$="P" THEN DISPLAY A
T(22,1):"PRINTER? PIO" :: AC
CEPT AT(22,10)SIZE(-18):P$ :
: OPEN #2:P$ :: D=2
200 LINPUT #1:A$
210 IF POS(A$,CHR$(201),3)-0
 THEN 260
220 LN=ASC(SEG$(A$,1,1))6
+ASC(SEG$(A$.2.1)):: P=3
230 X=POS(A$,CHR$(201),P)::
IF X=0 THEN 260 :: LREF=ASC(
SEG$(A$,X+1,1))6+ASC(SEG$
(A$,X+2,1)):: S=X-1
240 Z=ASC(SEG$(A$,S,1)):: G=
(POS("1341351761291691361482
37165",STR$(Z),1)+2)/3 :: IF
 Z<100 OR G<1 THEN S=S-1 ::
IF S<3 THEN G=1 ELSE 240
250 C$=STR$(LREF)"."&STR$(L
N)&STR$(G):: C(A)=VAL(C$)::
A=A+1 :: P=X+3 :: C$="" :: G
OTO 230
260 P=3 :: C$="" :: IF EOF(1
)THEN CLOSE #1 ELSE 200
270 DISPLAY AT(24,1):"SORTIN
```

IMAGINATIVE PROGRAMMING

by Jim Peterson

Most of the fun (and frustration) of programming is in figuring out a way to do something you haven't done before. Self-taught programmers often devise some highly unconventional methods of solving problems, and this example contains some doozies!

This is a greatly improved version of a program which appeared in Tips from the Tigercub. It is used to make a list of all program lines to which another line jumps, in order to avoid deleting those lines when modifying a program. Since it reads a program saved into a D/V 163 file by SAVE DSK (filename),MERGE it requires a disk drive.

Line 130 lists most of the Extended Basic statements which can direct a jump. In order to keep the array at less than 10 subscripts, for reasons to be explained later, I omitted the rarely used GO, BREAK and UNBREAK.

Line 140 dimensions an array to be used to store the records, and reads the DATA into another array. Line 150 asks for the filename of the MERGE format program to be examined. Note that the ACCEPT places the cursor after DSK, to make it plain that drive number and filename are to be entered. Line 160 opens the file and jumps over the error routine in 170. If the filename is not on the disk, etc., line 170 prints the error message and goes back to give you another chance. ON ERROR STOP prevents an erroneous I/O ERROR message if some later error should occur.

Line 180 obtains your choice of output, with a default of P, and rejects anything other than a single character P or S.

If printer output is selected, line 190 asks for printer designation, opens printer.

Line 200 reads in a program line, using LINPUT rather than INPUT so as to accept any and all characters. ASCII 201 is the token code that always precedes a line number reference, so line 210 searches for ASCII 201, starting at the 3rd character so as not to look for it in the program line number itself. If POS=0 there are no jumps from this line, so to 260.

Else, line 220 converts the tokenized 2-byte line number, of the line being examined, to its decimal form in LN, and sets P to start search for ASCII 201 at the 3rd byte. In line 230, X will be the position of the first ASCII 201, and the 2 bytes following it will be the line number jumped to, which is converted to its decimal equivelant LREF. The search for the

```
G" :: A-A-1 :: CALL LONGSHEL
LN(A,C())
280 IF D-0 THEN DISPLAY AT(1
2.1)ERASE ALL:"ANY KEY TO PA
USE"
290 FOR J=1 TO A :: A$=STR$(
C(J)):: X=POS(A$,".",1):: Y=
VAL(SEG$(A$.LEN(A$),1)):: A$
-SEG$(A$,1,LEN(A$)-1)
300 PRINT #D:SEG$(A$.1,X-1):
TAB(7):GO$(Y);" FROM ";TAB(2
1):SEG$(A$,X+1,LEN(A$))
310 CALL KEY(0,K,ST):: IF ST
=0 THEN 330
320 CALL KEY(0.K2,ST2):: IF
ST<1 THEN 320
330 NEXT J :: END
340 SUB LONGSHELLN(N,NN())
350 D=N
360 D=INT(D/3)+1 :: FOR I=1
TO N-D :: IF NN(I)<=NN(I+D)T
HEN 390 :: T-NN(I+D):: J-I
370 NN(J+D)=NN(J):: J=J-D ::
 IF J<1 THEN 380 :: IF T<NN(
J)THEN 370
380 NN(J+D)=T
390 NEXT I
400 IF D>1 THEN 360
410 SUBEND
```

statement directing the jump will begin at S, one byte to left of ASCII 201.

In line 240, Z is the ASCII code found at position S. The POS statement lists the 3-digit ASCII codes of the words listed in line 130, in the same sequence. If Z matches one of them, (G+2)/3 gives the subscript number of GO$ for that word. Otherwise, if Z<100 the ASCII is not a token, or if G=0 then (G+2)/3 is less than 1, so the next search starts one byte to the left. If S becomes less than 3, the token being searched for must be that for GO, BREAK, or UNBREAK, which are not in the POS statement, so we will use the subscript 1 for GOTO instead. Otherwise, go back and continue searching.

When the token is found, line 250 converts the "to" line number into a string followed by a period, followed by the string representation of the program line number followed by the string representation of the subscript number of GO$. For instance, if line 100 was 100 GOTO 1000, C$ would be 1000.1001, which can be converted into a valid decimal 1000.1001 .

Then, the search will continue starting at the 3rd byte after the last ASCII 201 we found, the add-on variable C$ is cancelled before being reused, and we go back to see if the program line contains another jump. If not, IF X=0 THEN 260, where we check EOF to see if the end of the file is reached, otherwise go back to read in the next program line, until finished.

Now, the data in C() is in program line number sequence, but we want to list it in "to" line number sequence. That's the reason for the weird method of decimal storage. Note that tacking the subscript number, from 1 to 9, onto the end of the decimal prevented the trailing zeros from being dropped. If C$="1000.100" then VAL(C$) would equal 1000.1, but VAL("1000.1001") equals 1000.1001.

So now, a simple sorting routine arranges the data into "to" line number sequence and, within that, into program line number sequence and even within that into subscript number sequence!

Finally, lines 290-330 convert the C() array back into a string so it can be taken apart into its three components and printed. Note that if printer output was not selected in line 180-190, D was not given a value and still equals 0 (its value in subroutine LONGSHELLN is not passed back to the main program) and PRINT #0 causes output to the screen.