* TI PPC NOTES *

V6N2, 1981

NEWSLETTER OF THE TI PROGRAMMABLE CALCULATOR CLUB.

9213 Lanham Severn Road, Lanham MD, 20801, USA.

The FAST MODE seems to stay as popular as ever. I keep receiving programs using this practical discovery. I even ventured myself into this mode and wrote my very first program, which I proudly present on page 5.

Of course, don't overlook Gary Holton's program on page 7. There, the FAST MODE really shines, to speed up usually slow sorting routines.

With respect to Paul Wright's Hyperbolic Functions in v6n1p11: According to Paul, some errors crept in. The mix up seems to be in the Greek letters I used. (It was all Greek to me ?) In the third paragraph, the first two lines contain the errors: ϕ, the Gudermannian of θ, abbreviated as gdθ; Sinhθ; Coshθ; Tanhθ; Tanh(θ/2); Cosh$^2\theta$;etc. As you can see, the changes are θ for ϕ; the addition of Coshθ; and the change to Cosh$^2\theta$ instead of Coshθ^2. (got it?)

One member produced an index and sent it to me. It turns out to be one of the most practical items I ever had since I started the newsletter, because I use it constantly to look up references to former articles. Unfortunately, that member's name and address got separated from the index, so that I am unable to tell you where to write to obtain a copy. So, if the one who produced this listing would be so kind and write me again, I will publish all the particulars in next issue.

F.van den Bogaard sends me a correction for the TI-57 LISTING ON A TI-59 program in v5n8p13: Alpha data register 18 contains now 5070.305109 and should contain 5250.305109. Thanks, Frans.

The HP PPC Eastern Conference will take place in Rockville, Maryland on March 28, 1981. Yours truly has been invited on an "open panel discussion." Richard Nelson will be there and so will be Jake Schwartz, as I understand it, THE driving force behind the HP PPC 's own module. I hope to learn a lot about how one makes his own module and about the possible pitfalls.

One of the members since February last year had the curious name of "Technical Calculator Programs, 5584 Tower Road, Riverside CA, 92506, USA. No first nor last name, which gave my sorting program on the TI-99/4 a fit. The mystery has been solved. I received a nice letter from Lonnie Mount, a registered Civil Engineer, who started the company by that name. Lonnie invites AOS or RPN programs for which he pays good money. The subject, for now, should be civil engineering. Lonnie plans to add other engineering branches later on. He is NOT interested in non-engineering programs. Thus, if you want to make some money from those Surveying, Hydraulics, Structural or Geology programs, here is your chance. It sure beats giving them away for nothing to the TI PPC NOTES or to PPX. And, of course, if you need a good program in those disciplines, just ask Lonnie for his catalog.

No news yet on our own module, except that about 150 members pledged a contribution. I will send all the suggestions for inclusion of certain routines to Lars Hedlund of the Swedish club, who will coordinate the whole effort. I will keep you posted on the progress we make.

Maurice E.T. Swinnen

IN THIS ISSUE:

DAY OF THE WEEK.- An alternate method to print the day of the week, using PGM 20 of the ML library, is demonstrated here in a program by Bill Beebe.
The routine produces an indirect address (SBR IND at steps 024, 025, 026). The table for each of the days follows the routine. Normally, such a method as presented here would not require as many steps as it seems. But one should keep in mind that the alpha data is not stored in a register, but in program memory itself. If the overall total memory is compared, this routine is slightly more efficient.
As can be seen, Bill had the problem of storing six 6-digit and one 8-digit numbers. He could have used THU for Thursday, but that would have caused confusion with TUE, for Tuesday. Since THUR comes before FRI, the first solution that would come to mind would be to add NOPs to each of the other days, making the program longer.
But his clever algorithm allows THUR to be the last word in the table, starting at step 079. This approach can be most useful for a large number of short words, say two or three characters long. On the average it can be as fast as indirect recall, and is much faster than having words packed in a data register, to be unpacked when the need arises.
The original DAY OF THE WEEK program was published in v5n9/10p18.
Instructions for this program are:
Enter the date in the form MMDD.YYYY and press A. Printer required.

```
122.1981     THUR
123.1981      FRI
124.1981      SAT
125.1981      SUN
126.1981      MON
127.1981      TUE
128.1981      WED

000  76  LBL
001  11   A
002  42  STD
003  04   04
004  36  PGM
005  20   20
006  14   D
007  75   -
008  06   6
009  75   -
010  69  OP
011  10   10
012  65   ×
013  07   7
014  95   =
015  65   ×
016  07   7
017  85   +
018  03   3
019  07   7
020  95   =
021  48  EXC
022  04   04
023  32  X:T
024  71  SBR
025  40  IND
026  04   04
027  69  OP
028  04   04
029  32  X:T
030  58  FIX
031  04   04
032  69  OP
033  06   06
034  22  INV
035  58  FIX
036  91  R/S
037  02   2
038  01   1
039  03   3
040  05   5
041  02   2
042  04   4
043  92  RTN
044  03   3
045  06   6
046  01   1
047  03   3
048  03   3
049  07   7
050  92  RTN
051  03   3
052  06   6
053  04   4
054  01   1
055  03   3
056  01   1
057  92  RTN
058  03   3
059  00   0
060  03   3
061  02   2
062  03   3
063  01   1
064  92  RTN
065  03   3
066  07   7
067  04   4
068  01   1
069  01   1
070  07   7
071  92  RTN
072  04   4
073  03   3
074  01   1
075  07   7
076  01   1
077  06   6
078  92  RTN
079  03   3
080  07   7
081  02   2
082  03   3
083  04   4
084  01   1
085  03   3
086  05   5
087  92  RTN
```

LOCAL CLUBS.- Gregory L. Stark, 14019 Cerise Ave, # 120, Hawthorne, CA 90250 would like to start a local club. Club members for that area who are interested in meeting with like-minded "calculator nuts", please write Gregory at the above address.

TI-99/4, MICROCOMPUTER.- If you own one of these marvelous 16-bit machines and would like to find out what the other owners are doing, here are two addresses where you can find out:
The TI 99/4 Home Computer Users Group, P.O. Box 95148, Oklahoma City, OK 73143. No dues. Newsletter at regular intervals. Sells software and accessories at rock bottom prices. Solicits user-generated software.
The 99'er Magazine, Emerald Valley Publishing Co.(owner Gary Kaplan) P.O. Box 5537, Eugene OR, 94705. Will publish the magazine soon at $ 15.00/year, 6 issues. Promises lots of software, programming aids. Solicits member participation. Offers two years at $ 28.00. (Canada and Mexico $ 18.00/year, Foreign surface $ 25.00/year, air mail at $ 40.00/year.)
Although this has nothing to do with the TI-59, I use the TI-99/4 to store the names and addresses of all the members and sometimes to print the mailing labels. And when the pressure becomes to acute, I play a game of chess against it. I still haven't beaten it in "advanced player" mode, but I don't give up hope.

PENPAL WANTED.- THOMAS BRETTINGER, SCHILLERSTRASSE 13, D-6452 HAMBURG 2, WEST GERMANY is interested in correspondance with calculator nut in the US. Thomas has a TI-59 with PC100B, and likes math, electronics, game programs and utility routines. If you remember v5n4/5p2, other TI Users Clubs, Thomas started a TI-users club, the IGPA, short for Interessengemeinschaft Programmierbaren Anwender. Thomas speaks, reads and writes English fluently.

MATHEMATICAL DIVERSIONS. Re-v5n3p16: Patrick Acosta of San Antonio, Texas, writes:
---------------------- About the pegboard problem: It turns out that every N X N pegboard has a solution, if N is NOT divisible by 2 or 3. Here is the proof: Write the numbers from 1 to N in the first row, then offset each succeeding row two places to the right. Thus:

1	2	3	4	5	6	7	8	......
N-1	N	1	2	3	4	5	6	......
N-3	N-2	N-1	N	1	2	3	4	......
N-5	N-4	N-3	N-2	N-1	N	1	2	......
.								
.								
.								
.								

Take any column, for example the 4th one, down that column, each succeeeding number is two less than the one above.(if the number is less than or equal to zero, add N) If N is coprime to N, that is, if their greatest common divisor is 1, then no number will be repeated. In number theory, a number will be repeated after k rows where 2k ≡ 0 modulo N, and if 2 is coprime to N, then k ≡ 0 modulo N, or k = N and there are N different numbers in that column.

For the right sloping diagonal, the difference is 1, so there are N different numbers along the diagonal.

Along the left sloping diagonal, each succeeding number is decreased by 3 and if 3 is coprime to N, no number will be repeated ever along the the longest diagonal.(*)

Notice that this does NOT show that if N is divisible by 2 or 3, there is no solution. So there may be a solution to an 8 X 8 pegboard, although I doubt it.

(*) Q.E.D. Isn't number theory wonderful?

THE TI-59 and babies? Nomi L. Burns, RN has written several programs on National Health
-------------------- Statistics Growth Charts for babies. She has clinical experience at the Texas Children Hospital in Houston, TX and at the Texas Tech's Health Sciences Hospital at Lubbock, TX in pediatric and neonatal nursing. Her programs permit you to find out if your child's growth is normal and where it falls within the range. She is willing to share these programs with our members. Please write to TOTS on TOP, P .O. BOX 64848, Lubbock TX, 79464. Mrs Burns is the wife of Robert T. Burns, former head of TI's TIPPP division.

GEOLOGY.- The United States Department of Interior, Geologic Survey has published three programs on the use of the TI-59 in geological survey:

Open-File report # 79-1662, TI-59 program for Interpretation of Refraction Seismic Data over up to four Dipping Layres, by David L. Campbell.

Open-File report # 79-1613, TI-59 program to calculate Theoretical MT Planewave Soundings over a Structure of up to 10 Horizontal Layers, by David L. Campbell.

Open-File report # 80-190, TI-59 program to calculate Theoretical Wenner and Schlumberger Vertical Electric Soundings over Structures of up to 10 Horizontal Layers, by Donald N. Haines and David L. Campbell.

Good documentation, listings, examples. Both authors are TI PPC Club members.

OPTICS.- EOSD, The Electro-Optical Systems Design Magazine, an Engineering Magazine of Electro-Optical and Laser Technology,(Cahners/Kiver Publishers, P.O. BOX 57, Denver CO, 80217) publishes every month a calculator column edited by Robert T. Pitlak. This month, in accordance with a newly announced policy, Robert reviews a couple of large programming systems on Ray Tracing. The first one is written by Gregory L. Stark, 14019 Cerise Ave, # 120, Hawthorne, CA, 90250. Gregory is a TI PPC Club member. His programming system comprises 1068 steps. The second program is by Walter R. Cook of THEON, Optical Systems Division, Box 166, White Creek NY, 12057. Walter is not a member yet. I suppose by writing to the authors you may obtain a copy of these large programming systems.

PERCENTAGE.- by Stuart Cox. This program allows the computation of five different types of percentage calculations in a natural, quick and easy manner, using only 102 program steps.

1. *Percentage: example $ 1.00 + 4 % = $ 1.04*
   Key 1, press +, key 4, press S, see 0.04, press =, see 1.04
2. *Change in percentage: example: $ 1.00 grows into $ 4.52. How many % increase is that? It is 352 %.*
   Key 1, press B, key 4.52, press R/S, see 452.
3. *Gross percentage: example: How much must be added to $ 4.00 so that the profit is 23.5 % of the total? $ 5.35 - 23.5 % = $ 4.00*
   Key 4, press +, key 23.5, press C, see 1.23, press =, see 5.23
4. *Net percentage: example: How much must be subtracted from $ 2.78 to give the amount to which 4 % was added to give $ 2.78 ? $ 2.67 + 4 % = $ 2.78*
   Key 2.78, press -, key 4, press D, see 0.11, press =, see 2.67
5. *Total percentage: example: $ 1.34 + 17.5 % - 13.25 % = $ 1.37*
   Key 17.5, press E, key 13.25 , press +/-, press R/S, see 1.93, then press X:T, key 1.34, press +, press X:T, press A, see 0.03, press =, see 1.37

```
000  76 LBL
001  11  A
002  53  (
003  24 CE
004  55  ÷
005  01  1
006  00  0
007  00  0
008  65  x
009  82 HIR
010  11  11
011  54  )
012  92 RTN
013  76 LBL
014  12  B
015  53  (
016  24 CE
017  85  +
018  32 X:T
019  00  0
020  54  )
021  92 RTN
022  53  (
023  53  (
024  24 CE
025  55  ÷
026  32 X:T
027  75  -
028  01  1
029  54  )
030  65  x
031  01  1
032  00  0
033  00  0
034  54  )
035  61 GTO
036  12  B
037  76 LBL
038  13  C
039  53  (
040  53  (
041  10 E'
042  75  -
043  02  2
044  54  )
045  53  (
046  94 +/-
047  35 1/X
048  75  -
049  01  1
050  54  )
051  65  x
052  82 HIR
053  11  11
054  54  )
055  92 RTN
056  76 LBL
057  14  D
058  53  (
059  10 E'
060  35 1/X
061  65  x
062  82 HIR
063  11  11
064  75  -
065  82 HIR
066  11  11
067  54  )
068  94 +/-
069  92 RTN
070  76 LBL
071  15  E
072  12  B
073  92 RTN
074  53  (
075  53  (
076  10 E'
077  65  x
078  32 X:T
079  10 E'
080  75  -
081  01  1
082  54  )
083  65  x
084  01  1
085  00  0
086  00  0
087  54  )
088  61 GTO
089  15  E
090  76 LBL
091  10 E'
092  53  (
093  24 CE
094  55  ÷
095  01  1
096  00  0
097  00  0
098  85  +
099  01  1
100  54  )
101  92 RTN
```

US PATENT # 4,153,937.- This patent was applied for on April 1, 1977 and granted on May 8, 1979. It covers a microprocessor system with higher order capabilities provided with two non-volatile memories which are read-only-memories in the disclosed embodiment.....for use as an electronic calculator... You guessed it, it is all about the TI-59. There is even a picture (drawing) that looks like a 59 without the TI emblem. It gives as the inventor Sidney W. Poland and the assignee Texas Instruments Inc. Dallas TX.

This document, which is in the public domain, of course, was sent to me by Dave Leising. It contains a total of 45 pages, 27 claims and 20 drawings. It provides fascinating reading. It contains, for example, tables upon tables with the hex addresses and the instruction words for all the chips. It tells you the exact locations of the firmware routines (variance, mean, standard deviation, sum plus, R to P, P to R, etc.)

What is most interesting, of course, is the explanation of the hierarchy registers. For HIR 20 it gives this explanation :"conditional return (second digit is ignored)" The rest of the HIR functions are identical to the ones we know already. No surprises there. There is also a drawing of the cross points of the keyboard. A lot of them are unused, which makes me wonder, if somebody would connect those unused ones to an external auxiliary keyboard, what functions could be unlocked. This is no idle speculation, as I once read in Byte a letter from someone relating an encounter with one of the designers of the TI-59. When asked why the SR-52 had a factorial key and the TI-59 had such a cumbersome way to obtain factorials through the ML module, the TI-man alledgedly answered:" It is in there, we just ran out of keyboard, so you have no way to get to it." An unused cross point might just bring the answer.

If you are interested, you might write the US Patent Office, Commissioner of Patents and Trade Marks, Washington DC 20231. State the patent number. The price is, unbelievably, $ 0.50. Yes, that is fifty cents, even if the patent contained a hundred pages. Their telephone number is 202-557-3158.

GEHEUGENVERLIES.- You must be used by now to the sometimes strange names I give to programs. I recall "Anwenderfreundlichkeit" and "Hoehenliniendiagram." This time the word has meaning in Flemish, or Dutch if you want, and it would be very difficult to translate it into English without resorting to a lot of verbosity. But why try? Just key in this program (no need to record it on a mag card, as it is so short) and press A. In a few seconds you will get the true meaning of the word and everything will become crystal clear to you. The closer to the beginning of the month of April this program can be run, the better, as its significance is rather closely related to spring rites. It reminds me of the old IBM proverb, on the wall of our computer room at work:"On a clear disk, you can seek forever."
Be sure to have the Master Library module in place.

```
000  00  0      019  93  .      038  06  6      057  54  )      076  28  LOG    095  07  7      114  00  00
001  00  0      020  05  5      039  03  3      058  65  ×      077  54  )      096  03  3      115  76  LBL
002  00  0      021  33  X²     040  65  ×      059  01  1      078  85  +      097  01  1      116  32  X:T
003  00  0      022  33  X²     041  03  3      060  00  0      079  53  (      098  65  ×      117  69  OP
004  00  0      023  59  INT    042  05  5      061  00  0      080  04  4      099  01  1      118  20  20
005  36  PGM    024  85  +      043  65  ×      062  54  )      081  03  3      100  07  7      119  06  6
006  02  02     025  08  8      044  06  6      063  84  OP*    082  65  ×      101  65  ×      120  32  X:T
007  71  SBR    026  54  )      045  54  )      064  00  00     083  03  3      102  04  4      121  43  RCL
008  02  02     027  65  ×      046  85  +      065  71  SBR    084  01  1      103  22  INV    122  00  00
009  39  39     028  03  3      047  01  1      066  32  X:T    085  54  )      104  28  LOG    123  67  EQ
010  09  9      029  22  INV    048  54  )      067  03  3      086  85  +      105  54  )      124  81  RST
011  00  0      030  28  LOG    049  65  ×      068  05  5      087  01  1      106  85  +      125  92  RTN
012  76  LBL    031  54  )      050  01  1      069  09  9      088  54  )      107  02  2      126  76  LBL
013  11  A      032  84  OP*    051  00  0      070  65  ×      089  84  OP*    108  54  )      127  81  RST
014  01  1      033  00  00     052  00  0      071  03  3      090  00  00     109  84  OP*    128  69  OP
015  42  STO    034  71  SBR    053  54  )      072  33  X²     091  71  SBR    110  00  00     129  00  00
016  00  00     035  32  X:T    054  85  +      073  65  ×      092  32  X:T    111  71  SBR    130  47  CMS
017  01  1      036  01  1      055  01  1      074  06  6      093  02  2      112  32  X:T    131  81  RST
018  03  3      037  00  0      056  07  7      075  22  INV    094  00  0      113  84  OP*
```

GAME OF SIMON.- Jeff Rosedale, by my account the youngest member but one of the most active ones, wrote this "Simon Says" program. It mimics the electronic game currently sold in this country. The calculator will randomly generate each number in a string of up to 90 numbers, all comprised between 1 and 4. The object is to memorize the whole string. The calculator will first show you one, single number. You repeat it. Then the TI-59 shows you two numbers. You also repeat them. Then you will be shown three, etc. up to 99. If you repeat an incorrect number, the calculator will flash the correct number of repetitions. No printer needed, but the ML module should be in place, as the program uses the random number generator of the module.

Instructions:

1. Key in the program and record on side 1 of a mag card.
2. Enter a seed between 0 and 1, for example .89765 and press E.
3. First number of the string will be pause-displayed. Then the calculator stops showing a zero.
4. Enter that number, not the zero, and press R/S.
5. Two numbers between 1 and 4 will be pause-displayed.
6. Enter the first number and press R/S. Enter the second one and press R/S.
7. The calculator will reward you by displaying three numbers in a row.
8. Enter each one separately, each one followed by R/S.
9. You now will be "tortured" by 4 numbers, etc. etc.
10. When you finally make a boo-boo, your correct repetitions will be flashed.
11. To begin a new game, no need to enter a seed, just press A.

NOTE: If you have masochistic tendencies and want to make the game harder on yourself, you can extend the range (1 to 4) by changing the high number at step 019 or the low number at step 021. But remember that the program uses direct addressing. At the end, the program leaves the calculator in 159.99 partition.

```
000  76  LBL    011  00  0      022  95  =      033  42  STO    044  69  OP     055  22  INV    066  69  OP
001  15  E      012  42  STO    023  59  INT    034  02  02     045  24  24     056  67  EQ     067  20  20
002  42  STO    013  00  00     024  72  ST*    035  01  1      046  97  DSZ    057  00  00     068  61  GTO
003  09  09     014  36  PGM    025  00  00     036  00  0      047  01  01     058  71  SBR    069  00  00
004  01  1      015  15  15     026  43  RCL    037  42  STO    048  00  00     059  69  OP     070  14  14
005  00  0      016  71  SBR    027  00  00     038  03  03     049  41  41     060  23  23     071  43  RCL
006  69  OP     017  38  DMS    028  75  -      039  42  STO    050  25  CLR    061  32  X:T    072  00  00
007  17  17     018  65  ×      029  09  9      040  04  04     051  91  R/S    062  97  DSZ    073  75  -
008  76  LBL    019  04  4      030  95  =      041  73  RC*    052  32  X:T    063  02  02     074  01  1
009  11  A      020  85  +      031  42  STO    042  04  04     053  73  RC*    064  00  00     075  00  0
010  01  1      021  01  1      032  01  01     043  66  PAU    054  03  03     065  51  51     076  95  =
                                                                                                077  16  A'
```

PERFECT NUMBER GENERATOR.- Dejan Ristanović wrote this program. Dejan is our only member in Belgrade, Yugoslavia. Dejan is an engineer and tells me that, besides his TI-59, he has an IBM-370 to play with. In his latest letter he "threatens" to try out a solution for our pegboard problem on that mighty IBM machine. We all know, of course, that this is called "cheating" and is rather "tacky." But then, how do I translate "tacky" into Serbo-Croatian ?

It has been said often that Pythagoras was fascinated with number theory. He really worshipped numbers. His fancy theories about numbers having special characters added little to the subject. But his work with prime numbers, perfect numbers and amicable numbers was really important.

Perfect numbers, the subject we deal with in this program, are numbers equal to the sum of their divisors, such as, for example 6 = 1 + 2 + 3. There are precious few perfect numbers below 10^{10}.

So, for the rigourously mathematically inclined members, here are a few definitions and theorems:

DEF: A number is perfect if and only if it is equal to the sum of its divisors.

THEOREM 1: If a number 2^n-1 is a prime number, then n is a prime number too.

Theorem 2: x is a perfect number if and only if $x = 2^{n-1}(2^n-1)$ and 2^n-1 is a prime number.

This program will generate the first six perfect numbers in 1 min 35 sec. The instructions to use it are:
Key in the program and record it on one side of a mag card.
Press GTO 5 LRN PGM 2 SBR 240 LRN RST R/S LRN LRN
Slide in the card and press R/S.
As you can observe, the program runs in FAST MODE. The FAST MODE is initialized here by placing PGM 2 SBR 240 in steps 005.... and following it by pressing RST R/S.
Then it is over-writen by the program itself.
By the way, the seventh perfect number, which this program does not generate and which has 12 digits is:
1 3 7 4 3 8 6 9 1 3 2 8 .

```
          6.
         28.
        496.
       8128.
   33550336.
 8589869056.
```

```
000  91 R/S     017  01  01     034  01  01     051  01  1      068  55  ÷      085  02  02     101  65  ×
001  06  6      018  15  15     035  32 X:T     052  42 STO     069  43 RCL     086  95  =      102  02  2
002  95  =      019  61 GTO     036  25 CLR     053  02  02     070  02  02     087  22 INV     103  45 Y×
003  66 PAU     020  00  00     037  03  3      054  04  4      071  95  =      088  59 INT     104  53  (
004  99 PRT     021  31  31     038  67  EQ     055  44 SUM     072  22 INV     089  67  EQ     105  43 RCL
005  01  1      022  02  2      039  00  00     056  02  02     073  59 INT     090  00  00     106  00  00
006  42 STO     023  45 Y×      040  22  22     057  43 RCL     074  29 CP      091  08  08     107  75  -
007  00  00     024  43 RCL     041  32 X:T     058  01  01     075  67  EQ     092  61 GTO     108  01  1
008  02  2      025  00  00     042  55  ÷      059  34 FX      076  00  00     093  00  00     109  95  =
009  44 SUM     026  75  -      043  03  3      060  32 X:T     077  08  08     094  54  54     110  66 PAU
010  00  00     027  01  1      044  95  =      061  43 RCL     078  02  2      095  87 IFF     111  99 PRT
011  01  1      028  95  =      045  22 INV     062  02  02     079  44 SUM     096  00  00     112  61 GTO
012  08  8      029  52 EE      046  59 INT     063  77  GE     080  02  02     097  00  00     113  00  00
013  32 X:T     030  22 INV     047  29 CP      064  00  00     081  43 RCL     098  22  22     114  08  08
014  43 RCL     031  86 STF     048  67  EQ     065  95  95     082  01  01     099  43 RCL     115  25 CLR
015  00  00     032  00  00     049  00  00     066  43 RCL     083  55  ÷      100  01  01     116  81 RST
016  77  GE     033  42 STO     050  08  08     067  01  01     084  43 RCL
```

STOCK AND OPTION ANALYSIS.- For those among you who are interested in the stock market: Datalab Inc., 3624 Science Center, Philadelphia, PA, 19104, tel. (215) 667-1640 has produced a special module, called Options MOD-1, that computes put and call model prices, implied volatilities, over/under valuation, hedge ratios, probability of exercise, Black/Scoles for calls, Parkinson for puts, etc. The module alone costs $ 265.00. That price includes the module, documentation, a six-month subscription to Options Analyst News or a three-month ditto to Market Makers Option Analyst News. Also included is a special keyboard overlay.

They have other products that might interest you, so, please, write them directly. And mention the TI PPC NOTES as the source, please.

I cannot vouch for the quality of the above, as I am not a stock market specialist. The information is only given so that you are aware of its existence.

ALPHABETIZING WITH NUMERICAL DESCRIPTORS.- *Gary Holton, of Godfrey, Illinois, wrote this FAST MODE program. Reversing the algorithm of Henrik Tjernberg's Sorting with Despriptors, v5n7p16, and using the techniques of Richard Snow's Fast Alpha Sort, v5n9/10p13, this program will alphabetize up to 89 five-letter words with integer descriptors of maximum three digits. The same Shell sort routine is used as in the original Alphabetical Sort. The sorted words are printed in sector 2, as the example shows, with their corresponding integers alongside in sector 3. It would be possible to device a routine that sorts 99 words in an OP 06 print routine, but this printout looks much better. The ML module should be in place, of course. Otherwise you won't get your calculator into the FAST MODE.*

Instructions:

1. *Key in the program of 240 steps in the turn-on partition and record bank 1.*
2. *Load bank 1 and press A to initialize the FAST MODE.*
3. *Load bank 1 again. A "1" will be printed.*
4. *Enter up to five characters, 10 digits, of alpha code and press R/S. Left-justified code is displayed.*
5. *Enter corresponding integer, max 3 digits long, and press R/S. The number of the next data point is displayed.*
6. *Go back to 4 and 5, until all your alpha code and descriptors are entered.*
7. *To correct a wrong entry: complete the data entry sequence, then enter the number of the data point you wish to correct as a negative value and press R/S. Continue entering in the normal fashion.*
8. *After all data has been entered press R/S once to start the sorting, after which a complete alphabetized printout will follow.*
   After printing, the program is ready to receive new data, provided you don't kill the FAST MODE by pressing RST.

```
        1.
  12 DA    26
  13DIG    16
  3D NA     4
  ANALY    17
  BRAIN    21
  CALEN     3
  DAY O    17
  FAALP    13
  FAMOI    14
  FAMON    22
  FRACT     5
  GUARD    10
  HOEHE     4
  LAPLA     4
  MARKE    24
  MICRO     4
  ML09     16
  MLSUP    21
  PRINT    18
  PROLI    18
  PROPU    20
  RADAR     4
  RADIA     4
  ROUNK    23
  SELEC     4
  SOLUT     3
  SPEED     6
  STRUC    17
  TIME      4
  TRANS     9
```

```
000 55 ÷      045 82 HIR    084 01 1      123 67 EQ     162 61 GTO    202 03 03
001 53 (      046 08 08     085 32 X:T    124 00 00     163 00 00     203 69 OP
002 34 ΓX     047 82 HIR    086 44 SUM    125 53 53     164 24 24     204 05 05
003 28 LOG    048 05 05     087 00 00     126 01 1      165 44 SUM    205 01 1
004 59 INT    049 55 ÷      088 43 RCL    127 82 HIR    166 00 00     206 82 HIR
005 75 -      050 02 2      089 00 00     128 05 05     167 32 X:T    207 35 35
006 04 4      051 95 =      090 77 GE     129 00 0      168 22 INV    208 82 HIR
007 54 )      052 59 INT    091 00 00     130 82 HIR    169 28 LOG    209 18 18
008 22 INV    053 82 HIR    092 61 61     131 04 04     170 65 ×      210 32 X:T
009 28 LOG    054 07 07     093 82 HIR    132 82 HIR    171 01 1      211 82 HIR
010 33 X²     055 82 HIR    094 15 15     133 15 15     172 00 0      212 15 15
011 95 =      056 55 55     095 32 X:T    134 42 STO    173 85 +      213 22 INV
012 59 INT    057 01 1      096 01 1      135 00 00     174 01 1      214 77 GE
013 85 +      058 85 +      097 85 +      136 73 RC*    175 85 +      215 01 01
014 66 PAU    059 42 STO    098 22 INV    137 00 00     176 28 LOG    216 39 39
015 91 R/S    060 00 00     099 77 GE     138 69 OP     177 59 INT    217 98 ADV
016 55 ÷      061 73 RC*    100 00 00     139 02 02     178 65 ×      218 29 CP
017 01 1      062 00 00     101 59 59     140 73 RC*    179 01 1      219 01 1
018 00 0      063 32 X:T    102 25 CLR    141 00 00     180 00 0      220 61 GTO
019 00 0      064 82 HIR    103 82 HIR    142 22 INV    181 00 0      221 00 00
020 00 0      065 17 17     104 18 18     143 59 INT    182 82 HIR    222 37 37
021 95 =      066 44 SUM    105 82 HIR    144 29 CP     183 44 44     223 00 0
022 72 ST*    067 00 00     106 05 05     145 67 EQ     184 02 2      224 00 0
023 00 00     068 73 RC*    107 82 HIR    146 01 01     185 75 -      225 00 0
024 69 OP     069 00 00     108 17 17     147 99 99     186 59 INT    226 76 LBL
025 20 20     070 77 GE     109 32 X:T    148 55 ÷      187 82 HIR    227 11 A
026 43 RCL    071 00 00     110 01 1      149 28 LOG    188 34 34     228 29 CP
027 00 00     072 93 93     111 03 3      150 59 INT    189 95 =      229 36 PGM
028 66 PAU    073 32 X:T    112 22 INV    151 42 STO    190 97 DSZ    230 02 02
029 91 R/S    074 72 ST*    113 77 GE     152 00 00     191 00 00     231 71 SBR
030 93 .      075 00 00     114 00 00     153 32 X:T    192 01 01     232 02 02
031 67 EQ     076 82 HIR    115 53 53     154 03 3      193 70 70     233 39 39
032 00 00     077 17 17     116 04 4      155 61 GTO    194 82 HIR    234 99 PRT
033 43 43     078 94 +/-    117 22 INV    156 01 01     195 14 14     235 01 1
034 77 GE     079 44 SUM    118 77 GE     157 65 65     196 75 -      236 25 CLR
035 00 00     080 00 00     119 00 00     158 69 OP     197 01 1      237 09 9
036 00 00     081 32 X:T    120 53 53     159 00 00     198 85 +      238 69 OP
037 50 I×I    082 72 ST*    121 01 1      160 22 INV    199 01 1      239 17 17
038 42 STO    083 00 00     122 22 INV    161 58 FIX    200 95 =
039 00 00                                               201 69 OP
040 61 GTO
041 00 00
042 28 28
043 43 RCL
044 00 00
```

BACKGAMMON.- *I received PPX program # 918217, 21 pages, by J. Brian Sladen. This 720 step program plays the game very well. It can be played with or without the printer. It will play at three levels: beginner's, intermediate or advanced. What is unique about this program is that when it is his turn to play, the user can ask the calculator what move it would make if their roles were reversed. He can also reverse the roles and play on from there. No such previleges are granted to the poor TI-59, however. This program is an excellent example of electronic decision making.*

FOR HIRMANIACS ONLY.- Have you ever hesitated to key in a program, just because it had so many HIRs in it? Rejoice,your suffering days are over. No more STO 82 BST BST DEL SST STO 39 BST BST DEL SST. Instead of those boring sequences you just load a card with nothing but code 82, just before you key in your program. Then you simply overwrite the not-needed 82s and leave the needed ones in place by pressing SST. Thus, the SST-key becomes, for all practical purposes, the HIR-key. Of course, if the number following the HIR is a two-digit number, you will still have to synthesize that one, but I guarantee you, your suffering will be much less now.

As with everything, you can do it the hard way or you can do it like people, the easy way. The hard way would be, that you key in STO 82 STO 82 until your face becomes as blue as the sky, after which you.... Well you know the procedure.

The easy way is like Manfred Gohly did it, over in Munich, West Germany.He wrote the program below and keyed it in. Then he started it with the A-key. Next he inserted a blank card into the slot, as far as it would go. After about 15 seconds the calculator pulled the card through, recording in the process bank 3 with nothing but HIRs.

You can perform the same feat by following these instructions. You now have a master card, which you can force into any bank by means of -N (1 +/-, or 2 +/-, etc.)

When I said, above, that "you will still have to synthesize double-digit numbers following a HIR", I mean that you can either do it the hard way or the easy way. The first method would again be STO NN BST BST DEL SST. The second way would be to learn by heart all the codes corresponding to the various keys. This way, if you need, for example, a 43 you just press RCL, or a 61 you key GTO.

```
000 76 LBL | 007 00 00 | 014 94 +/- | 021 02 2  | 028 02 2   | 035 97 DSZ | 041 03 3
001 11 A   | 008 08 8  | 015 52 EE  | 022 08 8  | 029 03 3   | 036 00 00  | 042 66 PAU
002 03 3   | 009 02 2  | 016 02 2   | 023 02 2  | 030 95 =   | 037 00 00  | 043 96 WRT
003 00 0   | 010 08 8  | 017 03 3   | 024 08 8  | 031 72 ST* | 038 31 31  | 044 91 R/S
004 42 STO | 011 02 2  | 018 75 -   | 025 02 2  | 032 01 01  | 039 22 INV |
005 01 01  | 012 08 8  | 019 93 .   | 026 08 8  | 033 69 OP  | 040 52 EE  |
006 42 STO | 013 02 2  | 020 08 8   | 027 52 EE | 034 21 21  |            |
```

Björn Gustavsson reworked a Danish program and came up with a shorter one. He published it in Programbiten 80-4, p 40. The idea is about the same as Manfred had. You start it by pressing A. It fills up bank 4 with HIRs. You may force-read that bank onto a card, banks 1 and 2, to make it more practical.

```
000 76 LBL | 006 53 (  | 012 08 8   | 018 02 2  | 024 54 )   | 030 54 )   | 036 31 31
001 11 A   | 007 53 (  | 013 02 2   | 019 08 8  | 025 65 ×   | 031 72 ST* | 037 42 STO
002 02 2   | 008 08 8  | 014 94 +/- | 020 02 2  | 026 01 1   | 032 00 00  | 038 00 00
003 09 9   | 009 02 2  | 015 75 -   | 021 08 8  | 027 52 EE  | 033 97 DSZ | 039 25 CLR
004 42 STO | 010 08 8  | 016 93 .   | 022 02 2  | 028 02 2   | 034 00 00  | 040 92 RTN
005 00 00  | 011 02 2  | 017 08 8   | 023 08 8  | 029 03 3   | 035 00 00  |
```

So, I told Bill Beebe about the idea over the phone. Bill lives about a thousand miles away from me, in Lilburn, Georgia. Bill likes a challenge. So, a couple of days later I received this one in the mail. Still the same idea, bank 4 filled with HIRs. But his program does it with even more step economy. Press A to start. After 10 seconds it stops with 959 in the display. Force-read onto a card, banks 1 and 2, then re-read it on another card, again with the force-read 1+/- and 2 +/-, but this time in the power-up partition 6 OP 17.

```
000 76 LBL | 004 42 STO | 008 02 2   | 012 00 0  | 016 95 =   | 020 00 00  | 024 69 OP
001 11 A   | 005 00 00  | 009 94 +/- | 013 55 -  | 017 72 ST* | 021 00 00  | 025 17 17
002 02 2   | 006 08 8   | 010 52 EE  | 014 09 9  | 018 00 00  | 022 17 17  | 026 91 R/S
003 09 9   | 007 93 .   | 011 03 3   | 015 09 9  | 019 97 DSZ | 023 25 CLR |
```

ERRATUM.- Re-v6n1p11. Paul B. Wright's address was incomplete. It should be 16860 Slover Ave, SP 93, Fontana , CA 92335.

HELP.- J. Batista, P.O. BOX 653, Potsdam, NY 13676, USA, would like to have a program for solving simultaneous non-linear equations for at least six variables. If you can help him, please write directly to the above address.

SELF-MODIFYING CODE FOR THE TI-58/59.- Byte, January 1981,pp 141-144 publishes an article by that name,authored by Ted Green, from Johns Hopkins University in Baltimore, MD. The idea was good, but as I said before and I have written to Byte on numerous occasions about it, their articles are not reviewed by somebody "at home" on calculators. If they had, they would have spotted the glaring blunder in this article: the author mistakenly associates register 99 with program steps 000, 001....007. If it had been printed only once, one could blaim the proverbial "printer". But there are even listings to prove it.The author repeatedly gives sequences such as : 10 OP 17 1234567891 STO 99 0 OP 17 GTO 000 LRN. That should be GTO 160, of course. The author forgets that there are 160 program steps that cannot be converted to data registers. The other 800 steps can, of course.

CODEBREAKER - SR-56. As promised, here is Don O'Grady's version of Mastermind for the SR-56. By the way, the listing is done with Bill Skillman's program SR-56 LISTING ON A TI-59. I will publish that one next time.

This program will both generate the secret code number and score your guess. Although that would be easy on a 52 or a 58/59, it is a "tour de force" on a 56.

Instructions:

1. *Key in the program.*
2. *Enter a seed, between o and 1. For example .97531, and STO 1.*
3. *Set program counter by pressing GTO 74.*
4. *Generate the secret code number by pressing R/S.*
5. *Enter your guess, consisting of four digits, PRECEDED BY A DECIMAL POINT. Press RST R/S.*
6. *The calculator will score your guess by displaying N.R in which N = the number of digits correctly placed and R = the number of correct digits in the wrong place.*

Repeat step 5 as often as needed, until your score becomes 4.0

For a new game, go to step 4. Do not initialize. → 59 32 x<>t

00	33	STO	15	32	X:T	30	00	0	44	04	4	58	05	5	72	07	7	86	29	INT
01	02	2	16	00	0	31	30	PROD	45	32	X:T	~~59~~	~~32~~	~~2~~	73	58	RTN	87	33	STO
02	04	4	17	37	X=T	32	03	3	46	33	STO	60	27	DSZ	74	49	FIX	88	01	1
03	33	STO	18	07	7	33	34	RCL	47	05	5	61	02	2	75	01	1	89	57	SUBR
04	00	0	19	01	1	34	03	3	48	34	RCL	62	09	9	76	34	RCL	90	00	0
05	01	1	20	04	4	35	29	INT	49	00	0	63	34	RCL	77	01	1	91	00	0
06	00	0	21	39	EXC	36	12	INV	50	37	X=T	64	04	4	78	64	×	92	32	X:T
07	30	PROD	22	00	0	37	35	SUM	51	05	5	65	33	STO	79	01	1	93	04	4
08	02	2	23	33	STO	38	03	3	52	04	4	66	00	0	80	92	.	94	12	INV
09	34	RCL	24	04	4	39	12	INV	53	92	.	67	27	DSZ	81	09	9	95	37	X=T
10	02	2	25	34	RCL	40	37	X=T	54	01	1	68	00	0	82	45	YX	96	07	7
11	29	INT	26	01	1	41	06	6	55	35	SUM	69	05	5	83	09	9	97	06	6
12	12	INV	27	33	STO	42	00	0	56	07	7	70	00	0	84	94	=	98	00	0
13	35	SUM	28	03	3	43	34	RCL	57	34	RCL	71	39	EXC	85	12	INV	99	58	RTN
14	02	2	29	01	1															

A LIFE CYCLE MODEL OF NEW PRODUCT PROFITABILITY.- Kenneth Deemer, Data Systems Design, Inc. Santa Clara, CA USA. Published in ACM (Association for Computing Machines),0-89791-024-9/80/0900-0150/0155.

A model, based on the product life cycle concept, considers the revenues and expenses associated with development, introduction, and marketing of new products. Key parameters may be varied individually to investigate the project's profitability under different scenarios. The product life cycle model may be programmed on a hand calculator (the TI-59) and used as a convenient first pass when evaluating new product proposals.

The article contains a good and clear explanation of the idea itself, a 465 step TI-58/59 program, mag card layout, and a worked out example.

The ACM grants permission to copy the article, if not done for commercial purposes. Clearly, our club is non-profit. So, if you send me $ 2.00 to defray copying and mailing costs, I'll send you a copy of this six-page article.

ALPHA + NUMERIC REGISTER LIST.- The author, Frèdèric De Mees, is a student in what in bilingual Belgium is called Informatica-Informatique, or what we call Computer Sciences.

He is also heavily bitten by the TI-59-bug, behavior which, according to the jargon used by my psychologists-friends at Walter Reed, should be positively reenforced." What a banana-flavored food pellet is to a monkey (the normal subject at Walter Reed) is the publication of one of your brainchildren to somebody who will dedicate the rest of his life to Informatique.

The program will list all non-zero data registers, up to 13 digits long. If it recognizes it as alpha code, it will also print the alpha on the same line. It will finally print the register number.

Data larger than 1EE15 or smaller than 1EE-3 will be printed in EE mode. If digits are missing it will also print a question mark.

Data smaller than 1EE10 and integer will be printed with its alpha equivalent.

Other advantages are: you can print all the registers with a different register number than the ones they are actually stored in. Secondly, there is no need to watch over the execution like a hawk in order to prevent paper waste. It runs between a start and a stop register number.

A has a little quirk. (nothing is perfect in this world) It sometimes prints a non-significant zero. But I doubt you will ever find out under which circumstances it does it.

```
      23.10001        24
     -143214.         25
    36373524.         26
    43.0011001        27
            0.        28
    43.0001001        29
    32500.4004        30
   -88621.0011        31
   -16241526.         32
```

```
 23.10001             24
 -143214.         BOB 25
 36373524.       STRI 26
 43.0011001           27
 43.0001001           29
 32500.4004044        30
 -88621.0011          31
 -16241526.      DICK 32
```

```
 23.10001             84
 -143214.         BOB 85
 36373524.       STRI 86
 43.0011001           87
 43.0001001           89
 32500.4004044        90
 -88621.0011          91
 -16241526.      DICK 92
```

The "mode d'emploi" or user instructions are:
Read in program. If you don't want alpha printing, set flag 0.
Enter a start register # and press A. Enter a stop register # and press R/S.
If you want to enter a different (fake) register number as the start register number to be printed now, quickly press R/S during the pause-display of the start register #.
Then enter your fake start register # and press R/S.

The top print out is a normal entry of 24 INV LIST. The next print out is obtained by pressing 24 A 32 R/S. The third print out was obtained by entering 24 A 32 R/S R/S (last R/S <u>during</u> the pause display) Then 84 R/S.

Frèdèric admonishes to "aller promener" after this, because the program takes about a minute per register.

Registers 0 through 63 are free to be used. The program uses registers 64 through 69 and all the HIR-levels.

If anybody wants a <u>real</u> challenge, rework this program so that it fits into 160 steps and all the data registers might be listed.

/SEE PROGRAM LISTING ON NEXT PAGE./

INTERPOLATION.- Re- v6n1p12. Fred Wiseogle wrote this short (19 step) program that accesses PGM 01 of the ML module. It does everything the Interpolation program in v6n1p12 does and more. It will also receive raw sets of data and calculate the best straight line through the points from which the interpolation can be obtained.

Instructions, in line with Fred Fullam's, are: Initialize, press A. When "0" appears enter 32 and press B. Then enter 3.16 EE 6 and press R/S. Next enter 100 and press B. Then enter 1.76 EE 6 and press R/S. Now find the interpolated data: enter 75 and press C. Enter 50 and press C, etc. Results are displayed only.

```
LBL A PGM 01 SBR CLR R/S LBL B X:T R/S Σ+ R/S B LBL C OP 14 R/S C
```

ALPHA + NUMERIC DATA REGISTER LIST.- Frédéric De Mees.

000	76	LBL	048	32	X:T	096	04	04	144	67	67	192	52	EE	239	67	EQ	286	59	INT
001	19	D'	049	07	7	097	73	RC*	145	52	EE	193	75	-	240	02	02	287	22	INV
002	53	(	050	69	OP	098	68	68	146	55	÷	194	59	INT	241	64	64	288	67	EQ
003	50	IXI	051	17	17	099	42	STO	147	52	EE	195	82	HIR	242	00	0	289	03	03
004	85	+	052	25	CLR	100	67	67	148	00	0	196	02	02	243	18	C'	290	16	16
005	01	1	053	32	X:T	101	50	IXI	149	00	0	197	95	=	244	61	GTO	291	73	RC*
006	85	+	054	42	STO	102	32	X:T	150	95	=	198	65	×	245	02	02	292	68	68
007	34	√X	055	68	68	103	00	0	151	28	LOG	199	82	HIR	246	28	28	293	50	IXI
008	23	LNX	056	75	-	104	67	EQ	152	58	FIX	200	13	13	247	32	X:T	294	55	÷
009	59	INT	057	91	R/S	105	03	03	153	00	00	201	22	INV	248	75	-	295	01	1
010	65	×	058	75	-	106	18	18	154	22	INV	202	28	LOG	249	52	EE	296	52	EE
011	02	2	059	01	1	107	01	1	155	52	EE	203	52	EE	250	95	=	297	01	1
012	54	)	060	95	=	108	52	EE	156	52	EE	204	95	=	251	29	CP	298	06	6
013	92	RTN	061	42	STO	109	01	1	157	58	FIX	205	42	STO	252	67	EQ	299	65	×
014	76	LBL	062	69	69	110	03	3	158	09	09	206	67	67	253	02	02	300	82	HIR
015	18	C'	063	43	RCL	111	22	INV	159	29	CP	207	01	1	254	57	57	301	37	37
016	50	IXI	064	68	68	112	77	GE	160	77	GE	208	82	HIR	255	25	CLR	302	01	1
017	32	X:T	065	66	PAU	113	02	02	161	01	01	209	53	53	256	35	1/X	303	52	EE
018	01	1	066	82	HIR	114	47	47	162	65	65	210	82	HIR	257	73	RC*	304	01	1
019	00	0	067	04	04	115	01	1	163	01	1	211	12	12	258	68	68	305	02	2
020	00	0	068	82	HIR	116	52	EE	164	94	+/-	212	22	INV	259	69	OP	306	95	=
021	49	PRD	069	14	14	117	03	3	165	82	HIR	213	52	EE	260	06	06	307	22	INV
022	66	66	070	55	÷	118	94	+/-	166	03	03	214	19	D'	261	61	GTO	308	59	INT
023	32	X:T	071	01	1	119	77	GE	167	87	IFF	215	18	C'	262	03	03	309	55	÷
024	44	SUM	072	00	0	120	02	02	168	05	05	216	29	CP	263	18	18	310	01	1
025	66	66	073	75	-	121	47	47	169	01	01	217	43	RCL	264	01	1	311	00	0
026	97	DSZ	074	69	OP	122	05	5	170	85	85	218	67	67	265	82	HIR	312	00	0
027	64	64	075	00	00	123	42	STO	171	29	CP	219	22	INV	266	37	37	313	95	=
028	00	00	076	42	STO	124	64	64	172	82	HIR	220	67	EQ	267	82	HIR	314	82	HIR
029	41	41	077	67	67	125	01	1	173	13	13	221	01	01	268	38	38	315	38	38
030	05	5	078	19	D'	126	42	STO	174	77	GE	222	67	67	269	87	IFF	316	69	OP
031	42	STO	079	52	EE	127	65	65	175	01	01	223	22	INV	270	00	00	317	05	05
032	64	64	080	02	2	128	25	CLR	176	85	85	224	87	IFF	271	03	03	318	01	1
033	25	CLR	081	48	EXC	129	42	STO	177	86	STF	225	05	05	272	16	16	319	44	SUM
034	48	EXC	082	67	67	130	66	66	178	05	05	226	01	01	273	01	1	320	68	68
035	66	66	083	95	=	131	73	RC*	179	04	4	227	67	67	274	00	0	321	82	HIR
036	84	OP*	084	65	×	132	68	68	180	00	0	228	05	5	275	22	INV	322	34	34
037	65	65	085	01	1	133	29	CP	181	18	C'	229	32	X:T	276	28	LOG	323	25	CLR
038	01	1	086	00	0	134	77	GE	182	61	GTO	230	43	RCL	277	32	X:T	324	97	DSZ
039	44	SUM	087	95	=	135	01	01	183	02	02	231	64	64	278	73	RC*	325	69	69
040	65	65	088	19	D'	136	40	40	184	16	16	232	67	EQ	279	68	68	326	00	00
041	92	RTN	089	44	SUM	137	02	2	185	43	RCL	233	02	02	280	50	IXI	327	68	68
042	76	LBL	090	67	67	138	00	0	186	67	67	234	64	64	281	77	GE	328	06	6
043	11	A	091	43	RCL	139	18	C'	187	55	÷	235	04	4	282	03	03	329	69	OP
044	22	INV	092	67	67	140	22	INV	188	82	HIR	236	32	X:T	283	16	16	330	17	17
045	57	ENG	093	22	INV	141	86	STF	189	13	13	237	43	RCL	284	29	CP	331	25	CLR
046	58	FIX	094	52	EE	142	05	05	190	22	INV	238	65	65	285	22	INV	332	92	RTN
047	09	09	095	69	OP	143	43	RCL	191	28	LOG									

KENDALL'S COEFFICIENT OF CONCORDANCE, W.- We have been neglecting the SR-52 lately. So, to make up for it, here is a nice statistics program for that calculator, written by Dick Blayney. On the next two pages you will also find two different explanations what Kendall's Coefficient of Concordance, W, is all about.

The user instructions are:

1. *Load side A of the mag card.*
2. *Enter number of judges, m and press 2nd A'. Display shows m.*
3. *Enter rank assigned by judge # 1 to entry # 1 and press RUN. Display shows R(1,1)*
4. *Enter rank assigned by judge # 2 to entry # 1 and press RUN. Display shows R(2,1)*
5. *Repeat for each judge: R(m,1)*
   When the last judge is entered, program automatically calculates the sum of the ranks for that entry. Display will show SUM R1.
6. *Repeat steps 3 through 5 for each entry.*
7. *Calculate coefficient W: press E. Display shows W.*

NOTES: 1. Max 17 entries and an infinite number of judges.
2. Entry with the lowest sum of ranks is first choice, i.e. the winner.

000	46	LBL	016	06	6	032	01	1	049	13	C	065	30	√X	081	43	RCL	097	00	0
001	16	A'	017	01	1	033	08	8	050	00	0	066	17	B'	082	00	0	098	85	+
002	47	CMS	018	44	SUM	034	43	RCL	051	42	STO	067	46	LBL	083	00	0	099	01	1
003	42	STO	019	00	0	035	01	1	052	01	1	068	15	E	084	55	÷	100	54	)
004	09	9	020	00	0	036	08	8	053	08	8	069	43	RCL	085	53	(	101	65	×
005	09	9	021	50	STF	037	75	-	054	36	IND	070	01	1	086	43	RCL	102	03	3
006	46	LBL	022	02	2	038	43	RCL	055	43	RCL	071	09	9	087	00	0	103	55	÷
007	17	B'	023	43	RCL	039	09	9	056	00	0	072	65	×	088	00	0	104	53	(
008	81	HLT	024	09	9	040	09	9	057	00	0	073	01	1	089	40	X²	105	43	RCL
009	42	STO	025	08	8	041	95	=	058	40	X²	074	02	2	090	75	-	106	00	0
010	09	9	026	36	IND	042	80	IF+	059	44	SUM	075	55	÷	091	01	1	107	00	0
011	08	8	027	44	SUM	043	13	C	060	01	1	076	43	RCL	092	95	=	108	75	-
012	60	IFF	028	00	0	044	43	RCL	061	09	9	077	09	9	093	75	-	109	01	1
013	02	2	029	00	0	045	09	9	062	22	INV	078	09	9	094	53	(	110	95	=
014	00	0	030	01	1	046	08	8	063	50	STF	079	40	X²	095	43	RCL	111	81	HLT
015	02	2	031	44	SUM	047	17	B'	064	02	2	080	55	÷	096	00	0			
						048	46	LBL												

SR-52

KENDALL'S COEFFICIENT OF CONCORDANCE, W.

If one wishes to determine the relationship among three or more sets of ranks, one rank could be selected and a Spearman rho coefficient computed between it and all of the others, and this process could then be continued until a rho coefficient has been obtained between each set of two ranks. Then these rho's could be averaged for an overall measure of relationship.

Kendall, though, has developed a technique and a statistic which makes all of this unnecessary. Suppose that five judges (m) rank the projects of ten individuals (N) in a judging contest, and we wish to determine the overall relationship among the ratings of the five judges. The rankings of these judges have been set up in Table 16.5. First the rankings by the five judges

TABLE 16.5. Calculation of the Coefficient of Concordance, the Data Consisting of the Ranking of Ten Projects by Five Judges

(1)	(2)					(3)	(4)	(5)
	Judges' Ranks							
Individual Project	1	2	3	4	5	Sum of Ranks	D	D^2
1	2	1	2	3	4	12	15.5	240.25
2	1	3	1	2	2	9	18.5	342.25
3	3	4	4	1	3	15	12.5	156.25
4	5	5	5	5	1	21	6.5	42.25
5	4	2	6	7	6	25	2.5	6.25
6	7	8	3	4	7	29	1.5	2.25
7	6	6	8	6	5	31	3.5	12.25
8	8	7	7	8	9	39	11.5	132.25
9	9	10	10	9	8	46	18.5	342.25
10	10	9	9	10	10	48	20.5	420.25
						$\Sigma = 275$		$\Sigma = 1696.50$

of each of the projects are summed and appear in column (3). Column (3) is summed to give the total sum of the ranks. This can be checked for the total sum of the ranks, as follows:

$$\text{Total sum of ranks} = \frac{mN(N+1)}{2}$$

$$= \frac{(5)(10)(11)}{2} = 275 \qquad (16.10)$$

If there were no relationship among the ranks, we should expect the sum of the ranks for each row to be equal. For this case the sum of each would be the average sum of ranks or $^{275}/_{10}$ which equals 27.5. We next obtain the difference of the sum of the ranks of each row from this mean and then square these differences. Then these squares are summed. This work appears in columns (4) and (5) of Table 16.5.

To compute W, we use the following formula:

$$W = \frac{12\Sigma D^2}{m^2(N)(N^2-1)} \qquad (16.11)$$

$$W = \frac{12(1696.5)}{(25)(10)(100-1)}$$

$$W = .82$$

Interpretation of W. The size of this coefficient of concordance indicates that there is high agreement among these five judges in the ranking of the ten projects. Perfect agreement is indicated by a $W = 1$ and lack of agreement by a $W = 0$. The significance of a coefficient of concordance may be tested by the use of tables developed by Kendall. For $m = 5$ and $N = 10$, we find that the W here computed is highly significant.

Sometimes we want to know the extent to which members of a set of m distinct rank orderings of N things tend to be similar. For example, in a beauty contest each of 7 judges ($m = 7$) gives a simple rank order of the 10 contestants ($N = 10$). How much do these rank orders tend to agree, or show "concordance"?

This problem is usually handled by application of Kendall's statistic, W, the "coefficient of concordance." As we shall see, the coefficient W is closely related to the average r_S among the m rank orders.

The coefficient W is computed by putting the data into a table with m rows and N columns. In the cell for column j and row k appears the rank number assigned to individual object j by judge k. Table 18.17.1 might show the data for the judges and the beauty contestants. It is quite clear that the judges did not agree perfectly in their rankings of these contestants. However, what should the column totals of ranks, T_j, have been if the judges had agreed exactly? If each judge had given exactly the same rank to the same girl, then one column should total to 7(1), another to 7(2), and so on, until the largest sum should be 7(10). On the other hand, suppose that there were complete disagreement among the judges, so that there was no tendency for high or low rankings to pile up in particular columns. Then we should expect each column sum to be about the same. In this example, the column sums of ranks are not identical, so that apparently some agreement exists, but neither are the sums as different as they should be when absolutely perfect agreement exists.

Table 18.17.1

	Contestants									
Judges	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	8	7	5	6	1	3	2	4	10	9
2	7	6	8	3	2	1	5	4	9	10
3	5	4	7	6	3	2	1	8	10	9
4	8	6	7	4	1	3	5	2	10	9
5	5	4	3	2	6	1	9	10	7	8
6	4	5	6	3	2	1	9	10	8	7
7	8	6	7	5	1	2	3	4	10	9
T_j	45	38	43	29	16	13	34	42	64	61

$$T = \frac{m(N)(N+1)}{2} = 385$$

This idea of the extent of variability among the respective sums of ranks is the basis for Kendall's W statistic. Basically,

$$W = \frac{\text{variance of rank sums}}{\text{maximum possible variance of rank sums}}.$$

Because the mean rank and the variance of the ranks each depend only on N and m, this reduces to

$$W = \left(\frac{12 \sum_j T_j^2}{m^2 N(N^2 - 1)}\right) - \frac{3(N+1)}{N-1}.$$

For the example, we find

$$\begin{aligned} W &= \left(\frac{12[(45)^2 + \cdots + (61)^2]}{49(10)(99)}\right) - \frac{3(11)}{9} \\ &= 4.28 - 3.66 \\ &= .62. \end{aligned}$$

There is apparently a moderately high degree of "concordance" among the judges, since the variance of the rank sums is 62 percent of the maximum possible. Note that by its definition, W cannot be negative, and its maximum value is 1.

PRINT CODE TABLE.- One Saturday morning I had the surprise visit of John Worthington and Emil Regelman, two noted contributors to the TI PPC NOTES. They make a living as scientists for the EPA, the Environmental Protection Agency. Once I was able to lure them away from the TI-99/4 and after they had beaten resoundingly the chess program on it, I got them to spit out their latest creations on the TI-59. As it is usually the case with too modest people, it turned out that they had a lot of programs they didn't think it worth to submit to the NOTES. Here is one of them. It proves again that there are at least a thousand ways to skin that cat. (see also v5n9/10p19)

As you may observe, the program runs in FAST MODE. The instructions are:

1. Slide the card, side 1, into the slot. Display shows a "1".
2. Press A to initialize the FAST MODE. Display shows a zero.
3. Slide the card, side 1, into the slot again.
   Printing starts right away. No need to press any key.

PRINT CODE TABLE

=00	-=20	.=40	↑=60
0=01	F=21	U=41	%=61
1=02	G=22	V=42	:=62
2=03	H=23	W=43	/=63
3=04	I=24	X=44	==64
4=05	J=25	Y=45	'=65
5=06	K=26	Z=46	×=66
6=07	L=27	+=47	x̄=67
7=10	M=30	×=50	²=70
8=11	N=31	*=51	?=71
9=12	O=32	√=52	÷=72
A=13	P=33	e=53	!=73
B=14	Q=34	•=54	Π=74
C=15	R=35	(=55	Δ=75
D=16	S=36	)=56	Π=76
E=17	T=37	,=57	Σ=77

```
000  00   0
001  00   0
002  00   0
003  76  LBL
004  11   A
005  36  PGM
006  02   02
007  71  SBR
008  02   02
009  39   39
010  09   9
011  00   0
012  22  INV
013  58  FIX
014  69  OP
015  00   00
016  25  CLR
017  52  EE
018  22  INV
019  52  EE
020  03   3
021  03   3
022  03   3
023  05   5
024  02   2
025  04   4
026  69  OP
027  01   01
028  03   3
029  01   1
030  03   3
031  07   7
032  00   0
033  00   0
034  01   1
035  05   5
036  03   3
037  02   2
038  69  OP
039  02   02
040  01   1
041  06   6
042  01   1
043  07   7
044  00   0
045  00   0
046  03   3
047  07   7
048  01   1
049  03   3
050  69  OP
051  03   03
052  01   1
053  04   4
054  02   2
055  07   7
056  01   1
057  07   7
058  00   0
059  00   0
060  00   0
061  00   0
062  69  OP
063  04   04
064  69  OP
065  05   05
066  69  OP
067  00   00
068  69  OP
069  05   05
070  02   2
071  03   3
072  05   5
073  09   9
074  09   9
075  09   9
076  02   2
077  94  +/-
079  82  HIR
079  05   05
080  01   1
081  07   7
082  06   6
083  04   4
084  00   0
085  01   1
086  00   0
087  08   8
088  82  HIR
089  06   06
090  03   3
091  07   7
092  06   6
093  04   4
094  00   0
095  04   4
096  00   0
097  03   3
098  82  HIR
099  07   07
100  05   5
101  07   7
102  06   6
103  04   4
104  00   0
105  06   6
106  00   0
107  08   8
108  82  HIR
109  08   08
110  01   1
111  52  EE
112  01   1
113  02   2
114  82  HIR
115  35   35
116  82  HIR
117  36   36
118  82  HIR
119  37   37
120  82  HIR
121  38   38
122  25  CLR
123  01   1
124  00   0
125  00   0
126  00   0
127  00   0
129  00   0
129  01   1
130  42  STO
131  01   01
132  03   3
133  00   0
134  00   0
135  00   0
136  00   0
137  09   9
138  03   3
139  42  STO
140  02   02
141  07   7
142  42  STO
143  03   03
144  02   2
145  42  STO
146  04   04
147  43  RCL
148  02   02
149  82  HIR
150  35   35
151  82  HIR
152  36   36
153  82  HIR
154  37   37
155  82  HIR
156  38   38
157  69  OP
158  05   05
159  43  RCL
160  01   01
161  82  HIR
162  35   35
163  82  HIR
164  36   36
165  82  HIR
166  37   37
167  82  HIR
168  38   38
169  69  OP
170  05   05
171  97  DSZ
172  03   03
173  01   01
174  59   59
175  07   7
176  42  STO
177  03   03
178  97  DSZ
179  04   04
180  01   01
181  47   47
182  98  ADV
183  98  ADV
184  98  ADV
185  98  ADV
186  25  CLR
187  91  R/S
```

DECIMAL TO FRACTION CONVERSION- Converting a fraction to a decimal is simple enough. Suppose you want to know the decimal corresponding to 9/16. Enter 9 DIV 16 =. But how do you do the inverse: what fraction is closest to, say, pi. This program by John Worthington and Emil Regelman does it within your specified tolerance. Enter the tolerance, from .1 through .0000000001 and press E. The smaller the tolerance, the longer the computation time, but the higher the precision. Enter the decimal and press A. Tolerance, numerator and denominator will be printed.

```
000  76  LBL
001  10  E'
002  43  RCL
003  05   05
004  76  LBL
005  15   E
006  22  INV
007  52  EE
008  32  X:T
009  03   3
010  07   7
011  03   3
012  02   2
013  02   2
014  07   7
015  04   4
016  00   0
017  69  OP
018  04   04
019  00   0
020  32  X:T
021  69  OP
022  06   06
023  42  STO
024  05   05
025  98  ADV
026  92  RTN
027  76  LBL
028  11   A
029  53   (
030  53   (
031  53   (
032  42  STO
033  00   00
034  75   -
035  59  INT
036  42  STO
037  03   03
038  54   )
039  65   x
040  69  OP
041  10   10
042  42  STO
043  08   08
044  54   )
045  42  STO
046  04   04
047  75   -
048  32  X:T
049  01   1
050  42  STO
051  02   02
052  00   0
053  67   EQ
054  99  PRT
055  29  CP
056  43  RCL
057  05   05
058  22  INV
059  67   EQ
060  95   =
061  93   .
062  01   1
063  33  X²
064  33  X²
065  15   E
066  76  LBL
067  95   =
068  54   )
069  32  X:T
070  00   0
071  48  EXC
072  00   00
073  99  PRT
074  00   0
075  32  X:T
076  22  INV
077  77   GE
078  76  LBL
079  53   (
080  43  RCL
081  04   04
082  85   +
083  43  RCL
084  05   05
085  54   )
086  32  X:T
087  01   1
088  22  INV
089  77   GE
090  77   GE
091  53   (
092  43  RCL
093  04   04
094  35  1/X
095  75   -
096  59  INT
097  42  STO
098  02   02
099  54   )
100  42  STO
101  01   01
102  29  CP
103  67   EQ
104  67   EQ
105  76  LBL
106  85   +
107  69  OP
108  20   20
109  53   (
110  43  RCL
111  00   00
112  55   ÷
113  43  RCL
114  01   01
115  85   +
116  93   .
117  05   5
118  54   )
119  53   (
120  59  INT
121  55   ÷
122  53   (
123  42  STO
124  06   06
125  65   x
126  43  RCL
127  02   02
128  85   +
129  43  RCL
130  00   00
131  54   )
132  42  STO
133  07   07
134  75   -
135  43  RCL
136  04   04
137  54   )
138  50  I×I
139  32  X:T
140  43  RCL
141  05   05
142  22  INV
143  77   GE
144  85   +
145  76  LBL
146  44  SUM
147  69  OP
148  00   00
149  69  OP
150  05   05
151  43  RCL
152  08   08
153  49  PRD
154  06   06
155  43  RCL
156  03   03
157  65   x
158  43  RCL
159  07   07
160  95   =
161  44  SUM
162  06   06
163  03   3
164  01   1
165  04   4
166  01   1
167  03   3
168  00   0
169  69  OP
170  04   04
171  43  RCL
172  06   06
173  69  OP
174  06   06
175  32  X:T
176  01   1
177  06   6
178  01   1
179  07   7
180  03   3
181  01   1
182  69  OP
183  04   04
184  43  RCL
185  07   07
186  69  OP
187  06   06
188  76  LBL
189  77   GE
190  98  ADV
191  92  RTN
192  76  LBL
193  99  PRT
194  43  RCL
195  03   03
196  99  PRT
197  76  LBL
198  67   EQ
199  25  CLR
200  01   1
201  76  LBL
202  76  LBL
203  42  STO
204  06   06
205  43  RCL
206  02   02
207  42  STO
208  07   07
209  61  GTO
210  44  SUM
211  76  LBL
212  12   B
213  43  RCL
214  06   06
215  32  X:T
216  43  RCL
217  07   07
218  92  RTN
219  76  LBL
220  13   C
221  71  SBR
222  02   02
223  13   13
224  53   (
225  32  X:T
226  55   ÷
227  32  X:T
228  54   )
229  92  RTN
```

Newcomer's Corner.- PETALS AROUND A ROSE.- The journal "Mathematics Teacher," Dec 1978, pp 753-753, published a puzzle under the name "the name of the game is Petals around the Rose." It has swept through entire school systems, offices, agencies, what have you.

The game is really quite simple. As a potentate of the rose, I know the secret that gives the answer for any roll of five dice. For example, the answer for figure 1b is 4. As a potentate I can tell you only three things:

1. The name of the game is Petals around the Rose, and the name is important.
2. The answer for any roll is always even.
3. I can also tell you the answer for any roll of the dice.

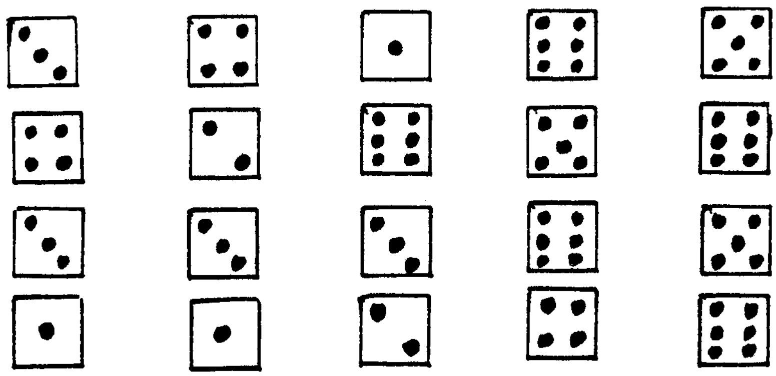

Fig 1, a = 6

Fig 1, b = 4

Fig 1, c = 10

Fig.1, d = 0

It takes some people days, or even weeks to see through this rather simple scheme. The object here is not to solve this puzzle, but to use it as a vehicle for showing how such a thing might be programmed. What if we had a program in which we entered all five digits corresponding to the roll of the dice and the program would give us each time the correct answer ? We could even expand the program, such that a roll of up to 9 dice was allowed, even with dice that had up to nine sides,if you can picture such an oddity in your mind.

So, we will have to write a verbal algorithm to tell to ourselves how exactly we are going to obtain the correct answer. But first, the secret: every die that has a center dot can be considered "a rose" and the outlying dots "the petals." So, a 5 will have 4 petals, a 3 will have 2 petals and all the other dice will be of no significance, as they do not have petals and as such do not contribute to the final result. Thus, the result will always be even, ranging from 0 through 20. (all fives)

What do we have to check for now, if our program has to examine each die? First if it is even we can stop the examination right away. How do you do that? You just divide it by 2 and check to see if the remainder is equal to zero. If "yes", the number was even, if "no" the number was odd.How do you find if the division has a remainder at all ? Simply, see if the result has a fractional part, by means of " INV 2nd INT." For example 23 DIV 2 = 11.5. If you now press INV 2nd INT you get .5. Try the same with 22 and see that the remainder will be zero. The first one is odd, the latter is even.

In a program,(not from the keyboard, for crying out loud) when you write "2nd CP", the only thing it will do is to set the t-register to zero. (From the keyboard, on the other hand, it will wipe out your entire program.) It will now be rather simple to program a sequence to see if the number is odd or even: (suppose the number is in the display, either entered from the keyboard or recalled from a register):

```
..... DIV  2  =  CP  INV INT  X=T  A'  GTO  B'
LBL A'...the number is even.    LBL B' .... the number is odd.
```

Now, only three odd numbers are of interest to us: 1, 3 and 5, and of those only 3 and 5 will contribute to the result. How much will they contribute: the number itself minus 1. So 3 will contribute 2 and 5 will contribute 4.(2 and 4 petals, remember)

I am sure by now you would be able to program this thing, if the face of each die was entered separately. You just check it if it is even or odd. If odd, you subtract

(over)

a 1 from it and sum the result into a register. After five entries you recall that register and voilà, the result.

But we said we could expand the program for up to nine dice. Well, come to think of it, we could accomodate even ten dice. Can't you enter ten digits in the display? Boy, you'll say, why can't he leave well enough alone. The program worked so nicely and was simple and straightforward. Now he has to mess it up by entering ten digits at a time, and we will have to peel off one at a time and check it. But, I will retort, consider the alternative: the poor user would have to enter each digit one at a time, followed by, say, pressing key A. That is an awful lot of key pressing and might lead to a lot of mistakes. "Nicer" would be to copy the roll of the ten dice into the display and just having to press key A once.

Now, for a peel off routine: There are several ingenious schemes to do that. Some of them peel from the front, others from the back. We will consider the simplest here, not the fastest: peeling off each digit from the back, that is the least-significant digit. To this end, we divide the entire number each time by 10, store the resulting integer in a separate register and store also the resulting fraction, multiplied by 10, in another register. The latter we check for "odd" or "even", then go back and divide the integer part again by 10, etc. until the entire ten digits have been peeled off. To show the principle, key in this sequence. Then store, for example 1234567891 into register 00 and press A. You will see the single digit "1". Press A again and you will see the "9" peeled off. Press A again and you will see the peeled off "8", etc.

LBL A RCL 00 DIV 10 = INV INT X 10 = STO 01 RCL 00 DIV 10 = INT STO 00 RCL 01 R/S

From all these elements we can write the entire program. One thing we should do first in that program, is to check if the entered number is zero. That will stop the execution right away. If it is not zero, we peel off each digit, check if it is odd or even. If odd, we subtract 1 from it and sum it into a register. At the very end we recall that register with the final result. A first cut of the program would look like this:

```
000  76 LBL
001  11  A
002  47 CMS
003  42 STO
004  00  00
005  76 LBL
006  43 RCL
007  43 RCL
008  00  00
009  29 CP
010  67  EQ
011  44 SUM
012  55  ÷
013  02  2
014  95  =
015  22 INV
016  59 INT
017  32 X:T
018  43 RCL
019  00  00
020  55  ÷
021  01  1
022  00  0
023  95  =
024  42 STO
025  03  03
026  22 INV
027  59 INT
028  42 STO
029  04  04
030  43 RCL
031  03  03
032  59 INT
033  42 STO
034  00  00
035  25 CLR
036  22 INV
037  67  EQ
038  42 STO
039  61 GTO
040  43 RCL
041  76 LBL
042  44 SUM
043  43 RCL
044  01  01
045  92 RTN
046  76 LBL
047  42 STO
048  43 RCL
049  04  04
050  65  ×
051  01  1
052  00  0
053  75  -
054  01  1
055  95  =
056  44 SUM
057  01  01
058  61 GTO
059  43 RCL
```

An experienced programmer, however, might see a lot of unnecessary steps in this program. It is nicely structured and easy to follow, but is is not optimized. This is then what Richard Snow did to my program: he enhanced it, shortened it and sped it up. It is not so easy to follow now, however. That is the price you pay for it: the more cryptic a program becomes, the more difficult it becomes to unravel it.

```
000  76 LBL
001  11  A
002  47 CMS
003  42 STO
004  00  00
005  76 LBL
006  43 RCL
007  43 RCL
008  00  00
009  29 CP
010  67  EQ
011  44 SUM
012  55  ÷
013  02  2
014  55  ÷
015  22 INV
016  59 INT
017  32 X:T
018  05  5
019  75  -
020  59 INT
021  42 STO
022  00  00
023  95  =
024  42 STO
025  04  04
026  25 CLR
027  67  EQ
028  43 RCL
029  43 RCL
030  04  04
031  65  ×
032  01  1
033  00  0
034  75  -
035  01  1
036  95  =
037  44 SUM
038  01  01
039  61 GTO
040  43 RCL
041  76 LBL
042  44 SUM
043  43 RCL
044  01  01
045  92 RTN
```

And finally Richard wrote his own version, taking advantage of all the tricks (not) in the book. This one is even shorter, faster but rather difficult to find the original algorithm in its lines. It is a nice example of extreme optimization.

```
000  76 LBL
001  11  A
002  29 CP
003  67  EQ
004  44 SUM
005  55  ÷
006  02  2
007  55  ÷
008  22 INV
009  59 INT
010  32 X:T
011  05  5
012  75  -
013  59 INT
014  42 STO
015  00  00
016  95  =
017  65  ×
018  09  9
019  65  ×
020  32 X:T
021  65  ×
022  02  2
023  95  =
024  59 INT
025  44 SUM
026  01  01
027  43 RCL
028  00  00
029  81 RST
030  76 LBL
031  44 SUM
032  00  0
033  48 EXC
034  01  01
035  92 RTN
```

Maurice