Notes on the back story of this letter:

I sent this *10-page* letter to ***Richard Nelson*** shortly after the previous one. The date is quite probably wrong: it wasn't stated in the letter proper, the envelope is missing, and there's some evidence that suggests it was written significantly earlier than the alleged Sept., 1980 date.

That aside, the letter includes a number of my materials, all of them intended for publication in the *PPC Calculator Journal*, namely:

(1) ***RF***, a short but very capable & fast general *root finder* routine for the **HP-41C**, which features an interactive prompting version *(RFP)* as well as a non-prompting one *(RPF)* callable from user's programs. Errors are automatically avoided if the derivative happens to be 0, as are infinite loops in case of non-real roots or nonconvergence. This routine was submitted for its inclusion in the *PPC ROM* (only 83 bytes) and of course it didn't make it.

(2) *A* ***Chess*** playing program for the **HP-67** which allows the user to play chess vs. the *HP-67*. The user conducts the *Black* king vs. the whole *White* army conducted by the *HP-67*. If the program manages to give *checkmate* in 6 moves or less, it wins; else, the user wins (being *stalemated* is also a win for the user).

(3) ***T***, a revision of the *Arithmetic Teacher* program featured in the *Standard Applications Pac* included with every **HP-41C**. My revision works the same as the original but it's 120 lines/32 registers (vs. 163/52) so it saves as many as *20 registers* for other uses and fits in a single mag card. I naively expected the programs in the *Standard Pac* to be optimized examples demonstrating efficient programming techniques but it seems I was wrong.

(4) Inputs to the ***FEEDBACK, 41C NOTES*** and/or ***ROUTINES*** sections, including my various optimizations and corrections to routines proposed for the *PPC ROM* by diverse authors, e.g.: to *Improved Synthetic Key Assignments* by *Tom Cadwallader*, or to a routine by *William C. Wickes* which I discovered it gave erroneous results for a range of arguments.

(5) A **PROGRAMMING TIP** showing in full detail how to very easily create on the fly a program line containing any arbitrary text (synthetic characters and all) using just **`STO b`** and **`RCL b`**, no **`CODE`** or similar cumbersome programs needed. A full example is given.

Finally, I comment on the proposed inclusion in the *PPC ROM* of a multi-function plotting program by *Jake Schwartz* that would take some 987 bytes, i.e. about 25% of the *ROM*.

Valentin Albillo, 07-08-2022

Richard Nelson
Editor, PPC Journal
2541 W. Camden Place
Santa Ana, CA 92704
U. S. A.

Valentin Albillo (4747)
Padre Rubio, 61 - 2º C
Madrid 29
SPAIN

Dear Richard:

I cannot resist the temptation of writing another submittal to the PPC Journal. It must be some kind of PPCmania, you know; as soon as I have some free time, I spend most of it writing something for its publication in my beloved PPCJ , although I know you'd never published anything contributed by - poor Spanish member 4747 (that's me). Well, now seriously, here included are :

-several inputs for the columns "FEEDBACK" , "41c NOTES" , "TIPS"
-one input for the PPC Custom ROM : a very short & fast root finder
-one 67/97 program: A CHESS PLAYING PROGRAM

The root finder routine is included recorded on a magnetic card. The 67/97 program is not, as I have no 67/97 to record it, and a 41c version will not work on an actual 67/97. Several comments about the routines included:

- the programming tip shows how to synthesize any text, composed of any characters as a line in a program, making use of just STO b, RCL b, in a bugless machine. No "CODE", "DECODE" programs, no bugs, just STO b, RCL b. This is a part of a future article regarding the properties & applications of STO, RCL b. The article (to be contributed in the future) shows also how to break PRIVATE using only STO, RCL b, as I told you in my previous letter.

- the SYNTHETIC KEY ASSIGNMENTS program, which you think may be included in the PPC Custom ROM may be greatly improved. Here included is a simple modification that saves 19 bytes at no cost, and quite a lot of initialization time. If the WSTS option is deleted (see the reasons), another 12 bytes are saved. I have now a totally different version of this program that combines all the properties of "CODE", "DECODE" , "UNBLD", "REBLD" (Wickes) & "SYNTHETIC KEY ASSIGNMENTS" (Cadwallader) onto a single magnetic card, thus being much shorter, and certainly faster. However, the version is not complete to the moment, as I find a little inconvenient to look up OCTAL-HEXADECIMAL conversion tables, because all inputs/outputs to my version are octal numbers, rather than decimal or hexadecimal values.

- the input for the PPC Custom ROM is a numerical root finder. I think that such a routine is needed in many applications (math, engineering, finance, etc.) and so I have worked out a suitable one. I have carefully read the article on the HP JOURNAL about the solve key and decided to provide a method which, being very short (to avoid taking up excessive space in the Custom ROM) has most advantages of SOLVE, and it is nearly double fast. I didn't try to write a very convolute routine but a short, fast, general purpose one, as I think all routines included in the PPC Custom ROM should be. Despite its simplicity you'll find harder to beat its speed and efficiency. Try the given examples.

-the 67/97 program is my response to Joseph O. Holmes (3673), who asked for more 67/97 material, specially "non-mastermind" type programs. This one plays chess, and it is very hard to beat.

Only a final comment re: MULTI-FUNCTION PLOTTING by Jake Schwartz (V7 N1 P29)

Jake suggests the possibility of including its multi-function plotting in the Custom ROM. I disagree. Consider that most PPC members don't own a printer. What should they have to do with this program ? . Of course, PPC members who have a printer deserve some attention, but this program is about 987 bytes long, this is almost 1K of memory ! It is too much ! And, do you think that it is very useful to plot more than 3 curves at once ? Besides, the program is very slow. It takes a lot of time to work. Please, don't take these lines as a hard criticism to the program, I can't do it better, I only express my opinion that taking up 1K of memory of the PPC Custom ROM to include a program only useful to those members with a printer, and which also need to plot 3 or more functions at once, and they need to do it at maximum resolution, even if it takes 2 days to do it, is quite a specialized application, don't you believe ? Please, stop joking and look for short (or long) general purpose routines which are useful to all members

- this is an input to your column "41c NOTES"

- - - - - - - - - - - - - - - - - - -

REVISION OF ARITMETIC TEACHER PROGRAM , HP-41c STANDARD APP.

Here is an improved version of the program "Arithmetic Teacher" included in the Standard Applications Pac. The program runs following the same guidelines, only the initial seed is previously stored in R00 , and the name of the program is "T"

```
01 LBL"+"      41 FRC         82 1
02 +           42 STO 00      83 ST- 06
03 STO 04      43 SQRT        84 GTO 07
04 RTN         44 RCL 01      85 LBL 06
05 LBL"-"      45 ×           86 CF 05
06 STO 04      46 INT         87 "YES"
07 +           47 STO 02      88 AVIEW
08 STO 03      48 RCL 00      89 LBL 07
09 RTN         49 R-D         90 DSE 07
10 LBL"×"      50 FRC         91 GTO 03
11 ×           51 STO 00      92 CLA
12 STO 04      52 RCL 01      93 ARCL 06
13 RTN         53 ×           94 "-Ø% RIGHT"
14 LBL"/"      54 INT         95 AVIEW
15 STO 04      55 STO 03      96 FS? 06
16 ×           56 XEQ IND 05  97 RTN
17 STO 03      57 CLA         98 TONE 8
18 RTN         58 LBL 04      99 TONE 9
19 LBL"T"      59 ARCL 03    100 SIN
20 CF 29       60 ARCL 05    101 SIN
21 FIX 0       61 ARCL 02    102 TONE 8
22 LBL 01      62 "-=?"      103 TONE 8
23 "MAX?"      63 PROMPT     104 TONE 8
24 PROMPT      64 RCL 04     105 TONE 7
25 STO 01      65 X=Y?       106 TONE 8
26 ISG 01      66 GTO 06     107 TONE 8
27 LBL 02      67 "NO"       108 TONE 7
28 CF 05       68 AVIEW      109 TONE 8
29 CF 06       69 BEEP       110 TONE 9
30 10          70 FS?C 05    111 SIN
31 STO 07      71 GTO 05     112 SIN
32 STO 06      72 SF 05      113 TONE 9
33 "+, -,      73 SF 06      114 TONE 8
   ×, /?"      74 GTO 04     115 E/X
34 AON         75 LBL 05     116 TONE 8
35 PROMPT      76 ARCL 03    117 TONE 7
36 AOFF        77 ARCL 05    118 E/X
37 ASTO 05     78 ARCL 02    119 TONE 7
38 LBL 03      79 "-="       120 TONE 6
39 RCL 00      80 ARCL 04    121 .END.
40 R-D         81 AVIEW
```

-to use the program:

seed STO 00

(seed must be a positive, non-zero number)

XEQ "T" → MAX?

input the largest number to use:

N R/S → +,-,×,/?

select +,-,×, or / :

+ R/S → (a)+(b)=?
- R/S → (a)-(b)=?
× R/S → (a)×(b)=?
/ R/S → (a)/(b)=?

key in your answer:

answer R/S → YES or NO

after 10 problem had been solved, the score is displayed:

→(SCORE)% RIGHT

if you made no mistake, a tune is played.

Remark : the instructions are exactly the same as those in the Standard Pac program. All specifications are the same, and the same tune is played

COMMENTS : The program has been reduced from 163 lines (52 registers) to 120 lines (32 registers), thus as many as 20 registers of program memory are saved. In its present format the program exactly fits onto 1 magnetic card, so key it in exactly as it's written. A single byte added makes it neccessary to use a second mag card. Let the final END of program memory be the END of the program. SIZE is reduced from 010 to 008. The specifications are the same as the previous ones, except some prompting sounds have been changed. The tune played is exactly the same, the seed should be stored previously in R00, and the name of the program is changed to "T". Happy programming.

VALENTIN AIBILLO (4747)

-the following are some inputs to "FEEDBACK" or "41c - NOTES" or "ROUTINES" , I am not very sure in fact...

- - - - - - - - - - - - - - - - - -

Reference: HIGH RESOLUTION PLOTTING by Nathan Meyers (4795) see V7 N2 pp. 49-50

-steps 101, 103, 105, 107, 109, 111 can be changed to ST+ X all of them ; this saves 12 bytes and speeds up the process.

-the 11.017 at lines 53, 134, 152 may be stored in an unused register at the beginning, and recalled at proper times. This would save at least 6 bytes, and runs faster as well.

VALENTIN ALBILLO (4747)

- - - - - - - - - - - - - - - - - -

Reference: SYNTHETIC FUNCTION ROUTINES by Bill C. Wickes see V7 N3 pp.6-7

-the routine "MANT" does not work properly for an arbitrary number. In fact, most numbers X less than 1 give erroneous results.

For instance, for x=FRC(PI) "MANT" gives 1.4159265-00 , (it should be .141592654), which can't be used in most subsequent computations because of its -100 exponent.

VALENTIN ALBILLO (4747)

- - - - - - - - - - - - - - - - - -

Reference: IMPROVED SYNTHETIC KEY ASSIGNMENTS program by Tom Cadwallader (3502). See V7N3P3

-lines 13 to 22 (both included) may be shortened to a single text line: it should be a text composed of the following 5 characters:

- full man (HEX 01, DEC 01)
- box star (HEX 69, DEC 105)
- "mu" (HEX 0C, DEC 12)
- null (HEX 00, DEC 00)
- box star (HEX BF, DEC 191)

this text is easily created, and once it is synthesized as a line in the program, it saves 19 bytes (almost 3 reg) and saves much time of initialization (in fact, 5 sub - routine calls to LBL A are avoided).

-LBL C , the "write status option" , should be deleted. I don't think it is efficient to waste 12 bytes in the PPC Custom ROM in a function that is already in the Card Reader ROM, and that may be assigned to a key for execution.

-further refinements are possible. I will submit a much shorter version of this program very soon (I hope).

VALENTIN ALBILLO (4747)

- - - - - - - - - - - - - - - - - - - -

PROGRAMMING TIP : Do you want to create an arbitrary text as a line in your program ? Follow this example: (You'll need only RCL b, STO b assigned to keys)

Let us create the program line 05, which is to be a text of the following 5 characters:

- full man (HEX 01), box star (HEX 69), "mu" (HEX 0C), null (HEX 00), box star (HEX BF)

1º) In the desired line enter a normal text of the same number of characters as the text you wanto to create:
05 "ABCDE"

2º) BST, press SIN , ST+ IND 18

3º) PACK

4º) position yourself at the ST+ IND 18, switch to RUN mode (if you followed (2º) and (3º) strictly, you already were at line ST+ IND 18, of course)

5º) RCL b, go out of USER mode

6º) switch to PRGM, BST, (you should see SIN), delete the SIN

7º) PACK , switch to RUN mode

8º) STO b , switch to PRGM mode. You should see something like ST+ IND M (as it's the case in this example)

9º) Looking at the HEX TABLE (V6 N5 P-22,23), key into program memory the keys whose HEX codes are the same as the characters you want to create:

-in this example: full man = LBL 00
box star = FRC
"mu" = LBL 11
("null", a little problem) "mu" = LBL 11
box star = GTO 14

so, key in (you are at line ST+ IND M) : (first, go out of USER mode, of course): LBL 00, FRC, LBL 11, LBL 11, GTO 14

10º) In this example, the 4th character was a null. To simplify the process, the null was introduced as another "mu". Now, to create the null:

-switch to RUN mode, STO b, (SWITCH BACK TO PRGM) (you should see again the ST+ IND M) ; SST, SST, SST, SST (you should see the second LBL 11). Delete the LBL 11 you are seeing. Delete the ST+ IND 18 you are seeing now. SST and you should see your synthesized line.

11º) Delete the (in this example) 5 lines after your text: they are the A,B,C,D,E of the original text.

12º) Your text is created. Be careful not to make a mistake during the whole process. Remember, nulls are introduced as any single-byte instruction which is deleted later.

Simple, isn't it ? Only STO b, RCL b, no CODE, no DECODE, no bug, no nothing.

VALENTIN ALBILLO (4747)

- the following is an input to your column "ROM PROGRESS"

- -

After many years of owning several HP's programmable calculators, (HP-25, HP-67 & HP-41c), I have realized that several kind of subroutines appear very frequently enclosed into the body of another main program.

One of this frequent routines is a root finder program: it finds the roots (one root at a time) of a general equation f(x)=0. It appears in :

- mathematical problems: solve f(x)=0 (of course!)
- engineering: gear wheels (tanx - x = I)
- chemical problems: weak acid-base titrations
- financial problems: find IRR of an investment

and many others. This need is recognized even by HP: the new handheld calculator HP-34c includes a numerical root finder, SOLVE , among its built-in capabilities.

All this makes me think that a numerical root finder method should be implemented into the PPC Custom ROM . This would render all problems mentioned before to be considered as trivial programs. See examples. Now, - what kind of numerical root finder ? Anyone who believes that root finding is easy, should read the article "Personal calculator has key to solve any equation f(x)=0", published in the HP JOURNAL, December 1979, pp-20. This article gives an idea of the kind of troubles one gets into when trying to program a "foolproof" root finder.

After careful thought, I decided that a soft - ware version of the hardware SOLVE function on the 34c , would not be very efficient: the SOLVE function is almost foolproof, it takes many attention to the resources neccessary to cope with bad guesses, unconvenient f(x), noise error, etc , resulting in not very fast execution times. Besides, SOLVE always finds the root to 10 significant places, but most real world applications are exact to 2 or 3 places only, and finding to 10 places a root of an equa - tion whose numerical coefficients are reliable to 2 places is a waste of time, to say the least.

Taking all this into consideration, I decided to present the following numerical root finder for its inclu - sion into the PPC Custom ROM:

```
01 LBL"RF"      12 5Ø           23 XEQ IND ØØ   34 RND
02 "NAME?"      13 STO 0        24 X=Ø?         35 X=Ø?
03 AON          14 LBL ØØ       25 GTO Ø1       36 GTO Ø1
04 PROMPT       15 RCL N        26 ST- M        37 DSE 0
05 AOFF         16 1            27 RCL M        38 GTO ØØ
06 ASTO ØØ      17 D-R          28 X=Ø?         39 SF ØØ
07 "XØ?"        18 D-R          29 SIGN         40 LBL Ø1
08 PROMPT       19 +            30 /            41 RCL N
09 LBL"RFP"     20 XEQ IND ØØ   31 D-R          42 CLA
10 CF ØØ        21 STO M        32 D-R          43 END
11 STO N        22 RCL N        33 ST- N
```

The program is 43 lines long, only 83 bytes. It requires SIZE 001 , and uses synthetic functions STO M,N,O, RCL M, RCL N, ST- M,N, DSE O. The registers M,N,O are used as mere auxiliars, to help save 3 registers. If synthetic - functions are not tolerated, M,N,O should be changed to Ø1,Ø2,Ø3 respectively. This increases size to SIZE 004, the line 42 CLA should be deleted, and 8 bytes would be saved. The routine uses flag Ø and doesn't alter the angular mode, fix display or whatever except the alpha register, which is used, then cleared.

It is based in the well-known Newton's method:

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

where the derivative $f'(x_n)$ is computed as

$f'(x_n) = (f(x_n+k)-f(x_n))/k$, where $k = 3E{-}4$

Thus, the method requires only 2 evaluations of f(x) per iteration, and its rate of convergence is quadratic: the number of exact places almost doubles on every iteration.

The program is written so to be short and fast: it is faster than SOLVE in most cases. The speed depends most on your f(x). Two versions are provided: (both in the same p)

- interactive version : call "RF", and the program will prompt for the name of your function, and the initial guess , x_o, for the root, then proceed to compute the root and stop with the root in the display.
- programmable version : call "RFP" , which assumes that the NAME of your function is in register 00, the initial guess ,XØ, is in the display. Then computes the root and returns it to the X register.

WARNINGS : - if y'= 0, the program takes y'= 1, then resumes the execution. This results in that a local minimum or maximum will give no trouble: it will get out of there in most cases.

- the program will perform up to 50 iterations. If the root is found within this limit, it will be presented, and the flag 00 will be cleared. If it is not found to the desired accuracy within 50 loops, the best approximation should be displayed, and flag 00 will be set . This avoids a forever running. You should test flag 00 to see if it is set or clear after the program returns to your main program, and decide what to do (maybe the root displayed needs a little refinement, or maybe the equation has no roots in that zone). This simulates the conditional property of SOLVE.

- the accuracy depends on the FIX n display. The greater n, the better the accuracy. Most times the computed root will be more accurate than it seems. If you need 2 or 3 places only, set FIX 2. If you need full accuracy, set FIX 7. Of course, more accuracy requires more time to achieve.

- after the execution, NAME remains unaltered in register 00, and alpha register is cleared.

EXAMPLES : using FIX 7 in all of them:

$e^x - 5x + 3 = 0$; X_o=1 gives 1.4688293 (12 seconds)
X_o=2 gives 1.7437520 (12 seconds)

(they are nearby roots and are correct to 9 places)

$x^2 - 4x + 4 = 0$; Xo=1 gives 1.9999908 (21 seconds)

(it is a double root, x=2 ; f(1.9999908) is zero to 10 plcs)

$x^2 + 1 = 0$; Xo=0 gives -3.3380759 and flag 00 is set

(after 50 iterations no root was found ; flag 00 is set, and the last iterated is presented ; there are no real roots)

$e^x - 2 = 0$; Xo=1 gives 0.6931472 in 8 seconds
Xo=20 gives 0.6931472 in 40 seconds
Xo=90 gives 40.0080372 and flag 00 is set ; the root was not found after 50 iterations, because the initial guess was very far from the root, and the EXP(x) is very sharp. However, XEQ"RFP" will give now 0.6931472 starting from the 40.0080372. Be a little careful with your Xo.

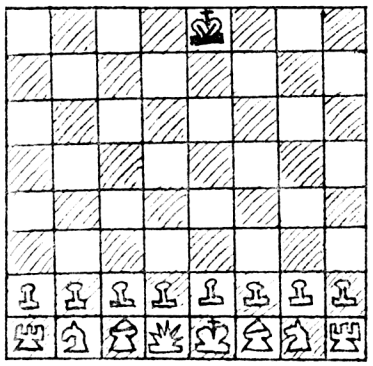
VALENTIN ALBILLO (4747)

P.S.: remember, this program is very short, but it should help in most cases. I could write a very cumbersome program which will take into account very special cases, but the ROM should have only general purpose routines, not complete courses on numerical analysis or whatever.

67/97 - A CHESS PLAYING PROGRAM

I totally agree with Joseph O. Holmes (3673) about the need for 67/97 programs. Not everyone in this Club owns a 41c. I HAVE a 41c (and a card reader, 3 modules, several pacs, etc), but I don't think that is good for the Club to dedicate all of its efforts to the new gadget. If PPC should grow, it should maintain its present membership as well, and it's quite possible to lose many members which don't find support for their models and don't understand either the many complexities of the 41c. So, please, stop a little with the repetitions of 41c material (for instance, article "MORE TONES" in V7 N3 P26 is almost a duplication of previous articles) and start with new and fresh 67/97, 34c, 33 materials.

This game program will allow you to play chess against any HP-67/97. Here the calculator plays the 16 white pieces, and you have only the black king. The initial position is the standard one given below. Calculator will try to play and checkmate in 6 moves or less: if it succeeds, it will be considered the winner. Of course, your goal is to play in such a way as to make a checkmate within 6 moves (or less) impossible. Also, you can try to force a stalemated position this is, a position in which you are neither in check nor able to make any legal move; in that case, you will be considered the winner too.



GAME CONVENTIONS : a) WHITE (HP): output of white's moves is performed accordingly to the following convention:

PIECE	CODE	PIECE	CODE
king ,K	6	bishop , B	3
queen,Q	5	knight ,Kt	2
rook ,R	4	pawn , P	1

Output is of the form XYZ.N, where:

-X = code of the piece White has just moved
-Y = code of the piece in whose column lays now the moved piece (up to 2 digits: the first one then specifies if it is a king side or queen side piece)
-Z = indicates in which square of the column rest the moved piece, counting upwards from the bottom.
-N = Number of White's moves already performed.

Examples: should White move a pawn to King's bishop 4 on its second move, the output would be:

1634.2 (= P-KB4 on the 2nd move)

if the output were 5627.4, this would be decoded so:

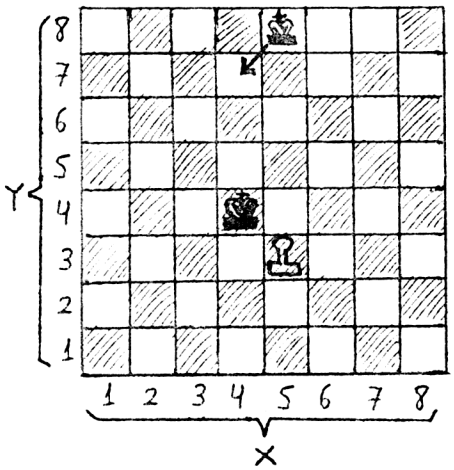
5 = stands for queen, Q
62 = 6 stands for King, K, and 2 for Knight, Kt
7 = the 7th square of the column
4 = the 4th move of White.

so, this means that White plays its Queen (5) to the 7th square (7) of its King (6) side knight (2) column on its 4th (4) move.

All in all, there are 3 kinds of output:

normal = XYZ.N , in DSP 1
check = -XYZ.N , in DSP 1
checkmate = -XYZ.OOOOON , in DSP 9

b) BLACK: your moves are input in quite a different way; to move your black king, you must input the XY coordinates of the square it moves to, following the



X-Y notation indicated in the figure. For instance, in the initial position your king is in 58. To move your king to the location indicated by the arrow, you must input: 47 R/S

If your king takes a white piece on its move, the procedure is the same as before, but you must change the sign of your input. To take the pawn in the figure, input -53 R/S.

To run the program, you will need to have the following data stored: (make a data card with them)

P.regis :	0	1	2	3	4	5	6	7	8	9
contents:	164	524	527	154	557	525	3634	537	155	336

P.regis :	A	B	C	D	E	I
contents:	1523	322	334	10	0.8	0

S.regis :	0	1	2	3	4	5	6	7	8
contents:	3525	5527	565	325	0	346	3534	567	5627

S.regis :	9
contents:	353

-This data card is needed only when loading the program.

WARNINGS : illegal moves of yourself can't be corrected, and a re-start would be needed.

-program does not always indicate check status. These situations are infrequent, however.

INSTRUCTIONS : load program card & data card

1) start the game: press A → White's first move
2) input your answer: XY R/S → White's 2nd move
   repeat (2) until HP checkmates you. If you reach the move number 6 without being checkmated, you have won. If HP checkmates you on its 6th move (or less), HP has won.
3) for another game, go to (1)

EXAMPLE : let's have a game. I'll play as bad as possible:

A → 164.1 (=P-K4), I move to 68: 68 R/S → 524.2 (=Q-Kt4)
67 R/S → 557.3 (=Q-Q7)
68 R/S → 1523.4 (=P-QKt3)
78 R/S → -334.5 (=B-B4)
A check !
68 R/S → -322.0000006
this is, B-Kt2.
Checkmate . I'VE LOST

- This is finished. HP has checkmated my king on its 6th move, so I have lost. I played very badly. Perhaps you will do better.

VALENTIN ALBILLO (4747)

```
001 LBL A   31 25 11
002 DSP 1      23 01
003 0          00
004 ST I       35 33
005 RCL 0      34 00
006 GSB 2   31 22 02
007 RCL 1      34 01
008 GSB 7   31 22 07
009 INT        31 83
010 6          06
011 X=Y        32 51
012 GTO C      22 13
013 RCL 2      34 02
014 GSB 7   31 22 07
015 FRAC       32 83
016 RCL E      34 15
017 X=Y        32 51
018 GTO D      22 14
019 RCL 3      34 03
020 GSB 2   31 22 02
021 2          02
022 6          06
023 X≠Y        32 61
024 GTO 0      22 00
025 RCL 4      34 04
026 LBL 8   31 25 08
027 GSB 2   31 22 02
028 RCL 5      34 05
029 LBL 9   31 25 09
030 CHS        42
031 DSP 9      23 09
032 ISZ        31 34
033 RC I       35 34
034 EEX        43
035 7          07
036 /          81
037 -          51
038 RTN        35 22
039 LBL 0   31 25 00
040 GSB 6   31 22 06
041 X≠Y        32 61
042 GTO 0      22 00
043 RCL 6      34 06
044 GSB 2   31 22 02
045 GTO 5      22 05
046 LBL 0   31 25 00
047 GSB 6   31 22 06
048 X≠Y        32 61
049 GTO 0      22 00
050 RCL 6      34 06
051 CHS        42
052 GSB 2   31 22 02
053 5          05
054 6          06
055 X≠Y        32 61
056 GTO 5      22 05
057 RCL 8      34 08
058 GTO 9      22 09
059 LBL 0   31 25 00
060 GSB 6   31 22 06
061 X≠Y        32 61
062 GTO 0      22 00
063 P()S       31 42
064 RCL 0      34 00
065 P()S       31 42
066 GSB 2   31 22 02
067 LBL 4   31 25 04
068 RCL 4      34 04
069 GTO 9      22 09
070 LBL 0   31 25 00
071 Rdown      35 53
072 1          01
073 5          05
074 X≠Y        32 61
075 GTO 0      22 00
076 P()S       31 42
077 RCL 1      34 01
078 P()S       31 42
079 GTO 8      22 08
080 LBL 0   31 25 00
081 P()S       31 42
082 RCL 2      34 02
083 P()S       31 42
084 GTO 8      22 08
085 LBL D   31 25 14
086 P()S       31 42
087 RCL 3      34 03
088 P()S       31 42
089 GSB 2   31 22 02
090 1          01
091 8          08
092 X≠Y        32 61
093 GTO 0      22 00
094 RCL 9      34 09
095 CHS        42
096 GTO 3      22 03
097 LBL 0   31 25 00
098 GSB 6   31 22 06
099 X≠Y        32 61
100 GTO 0      22 00
101 P()S       31 42
102 RCL 5      34 05
103 P()S       31 42
104 GTO 3      22 03
105 LBL 0   31 25 00
106 GSB 6   31 22 06
107 X≠Y        32 61
108 GTO 4      22 04
109 RCL 9      34 09
110 GSB 2   31 22 02
111 2          02
112 8          08
113 X≠Y        32 61
114 GTO 4      22 04
115 RCL 2      34 02
116 GTO 9      22 09
117 LBL C   31 25 13
118 RCL 4      34 04
119 GSB 7   31 22 07
120 FRAC       32 83
121 RCL E      34 15
122 X=Y        32 51
123 GTO B      22 12
124 RCL 3      34 03
125 GSB 2   31 22 02
126 6          06
127 6          06
128 X≠Y        32 61
129 GTO 0      22 00
130 P()S       31 42
131 RCL 6      34 06
132 P()S       31 42
133 GSB 2   31 22 02
134 LBL 5   31 25 05
135 RCL 7      34 07
136 GTO 9      22 09
137 LBL 0   31 25 00
138 GSB 6   31 22 06
139 X≠Y        32 61
140 GTO 0      22 00
141 P()S       31 42
142 RCL 7      34 07
143 P()S       31 42
144 GTO 8      22 08
145 LBL 0   31 25 00
146 Rdown      35 53
147 X)0        31 81
148 GTO 0      22 00
149 P()S       31 42
150 RCL 9      34 09
```

```
151 P()S        31 42
152 GTO 9       22 09
153 LBL 0    31 25 00
154 P()S        31 42
155 RCL 8       34 08
156 P()S        31 42
157 GTO 8       22 08
158 LBL B    31 25 12
159 RCL A       34 11
160 GSB 2    31 22 02
161 6              06
162 8              08
163 X≠Y         32 61
164 GTO 0       22 00
165 RCL B       34 12
166 GTO 3       22 03
167 LBL 0    31 25 00
168 GSB 6    31 22 06
169 X≠Y         32 61
170 GTO 0       22 00
171 RCL C       34 13
172 CHS            42
173 GSB 2    31 22 02
174 6              06
175 8              08
176 X=Y         32 51
177 GTO 5       22 05
178 RCL B       34 12
179 GTO 9       22 09
180 LBL 0    31 25 00
181 RCL B       34 12
182 CHS            42
183 LBL 3    31 25 03
184 GSB 2    31 22 02
185 RCL 2       34 02
186 GTO 9       22 09
187 LBL 6    31 25 06
188 RCL D       34 14
189 +              61
190 RTN         35 22
191 LBL 7    31 25 07
192 GSB 2    31 22 02
193 RCL D       34 14
194 /              81
195 RTN         35 22
196 LBL 2    31 25 02
197 X(0         31 71
198 SF 2     35 51 02
199 ABS         35 64
200 ISZ         31 34
201 RC I        35 34
202 RCL D       34 14
203 /              81
204 +              61
205 F? 2     35 71 02
206 CHS            42
207 R/S            84
208 RTN         35 22
```

Status: CF 0,1,2,3 / DEG / FIX / DSP 1

VALENTIN AIBILLO (4747)