HP Forum Archive 10

A little classical RPN challenge

*Message #1 Posted by **Ex-PPC member** on 6 Feb 2003, 9:44 a.m.*

Hi everybody, here's another little challenge, this time having to do with classical RPN, so most of you will be able to give it a think.

The Challenge:

1) Starting from this initial stack arrangement (where A, B are some arbitrary values), to reach this final arrangement, in a **minimum** number of steps:

```
T   -              T   -
Z   -              Z   B
Y   B      ->      Y   B
X   A              X   A
```

2) Same for this other case:

```
T   -              T   B
Z   -              Z   A
Y   B      ->      Y   B
X   A              X   A
```

You can use any functions and operations whatsoever available in your RPN calc, as long as you do it in a **minimum** number of steps, the solution asked for is the one that delivers the goods in as few steps as possible, notwithstanding concerns of efficiency, usability, etc. Remember, this is intended to be challenging and fun, not necessarily practical.

For the HP-15C & HP-11C, there are several solutions in as few as **three** steps for case (1) and **four** steps for case (2) above, and I'm confident those sames solutions are likely to also work on an HP-34C, HP-67, HP-41C, HP-42S and many other models, though I can't confirm it right now.

Flex your RPN muscles and give it a try ! The best answer posted within 96 hours gets (by e-mail) a good-quality scan of a very old, very rare, frankly bizarre HP calc brochure.

Well, for the 41c...

Message #2 Posted by ***Gene*** *on 6 Feb 2003, 9:53 a.m.,*
in response to message #1 by Ex-PPC member

Try this for #1

RCL Y X<>Y

2 steps

But you said the 11c / 15c variety, which don't include the RCL stack command. Continuing to think...but only have a 12c handy.

Re: Well, for the 41c... still

Message #3 Posted by ***Vieira, Luiz C.*** *on 6 Feb 2003, 10:34 a.m.,*
in response to message #2 by Gene

For case two:

RCL Y RCL Y

And we have it in two STEPS (four bytes, instead)

Hope this helps

Re: Well, for the 41c... still

Message #4 Posted by ***Ex-PPC member*** *on 6 Feb 2003, 11:42 a.m.,*
in response to message #3 by Vieira, Luiz C.

Thanks for the answers for the HP-41C (also valid for the HP42S) but of course, the challenge is for machines having classical RPN stack but no STO/RCL/X<> functions addressing stack registers (STO Y, RCL T, X<>Z, etc).

Else, it wouldn't be much of a challenge, indeed ...

See if you can find a minimum solution for machines such as the HP-10C/11C/15C ..., HP-25/27/29C ..., HP-34C, HP-67, HP-91/92/97 ..., etc, or even for the 41C/42S but avoiding the use of stack addressing for STO/RCL/X<>

ACK

Message #5 Posted by ***Vieira, Luiz C. (Brazil)*** *on 6 Feb 2003, 11:52 a.m.,*
in response to message #4 by Ex-PPC member

The answers would be restricted to two models, indeed.

Possible solutions for 11C

Message #6 Posted by ***Michael F. Coyle*** *on 6 Feb 2003, 5:27 p.m.,*
in response to message #4 by Ex-PPC member

For the first case (A B B -) I can't get it below four steps: [ENTER] [ENTER] [+] [R Down] I have another solution in four steps as well.

For the second case (A B A B) I have: [f CLEAR REG] [Sum+] [RCL Sum+] [RCL Sum+]

Looking forward to seein a better solution to #1, - Michael

But that would leave...Re: Possible solutions for 11C

Message #7 Posted by ***Gene*** *on 6 Feb 2003, 5:48 p.m.,*
in response to message #6 by Michael F. Coyle

THe stack after your (1) solution would be

A+B B B A

Do the rules allow the T level of the stack to be A+B?

I took the original instructions to imply it should hold a zero or whatever it held originally.

Good solution anyway! Gene

Re: But that would leave...Re: Possible solutions for 11C

Message #8 Posted by ***Michael F. Coyle*** *on 6 Feb 2003, 8:35 p.m.,*
in response to message #7 by Gene

Hi Gene,

It wasn't specified in the rules. I took the dashes "-" as "don't-cares" whose values were unimportant. But that's just my interpretation.

- Michael

Re: Possible solutions for 11C

Message #9 Posted by ***Vieira, Luiz C. (Brazil)*** *on 6 Feb 2003, 6:45 p.m.,*
in response to message #6 by Michael F. Coyle

Hi;

this was post a few hours ago.

But there is one problem: [CLEAR] [E+] also clears the stack registers contents in the HP11C and in many other calculators, so one should input A and B after clearing statistics registers with [f] [CLEAR] [E+].

Maybe the post is not so easy to view because it is not exactly in this branch of the original thread.

Re: Possible solutions for 11C

Message #10 Posted by ***Michael F. Coyle*** *on 6 Feb 2003, 8:47 p.m.,*
in response to message #9 by Vieira, Luiz C. (Brazil)

Hi Vieira,

I didn't read any of the replies before because I wanted to try it on my own without peeking at the answers, so to speak.

You're right about [CLEAR] [E+] clearing the stack. The 11C's [f CLEAR REGS] clears memory but leaves the stack untouched, so it's safe to use after the numbers are put on the stack.

Thanks for your reply.

- Michael

I forgot to mention...

Message #11 Posted by ***Vieira, Luiz C. (Brazil)*** *on 6 Feb 2003, 9:38 p.m.,*
in response to message #10 by Michael F. Coyle

... that your solution is better than mine because of this: [f][CLEAR][REG] is not destructive in relation to stack contents. The fact is that your solution works everytime, mine doesn't.

Thank you.

Best regards.

Re: A little classical RPN challenge

Message #12 Posted by ***Vieira, Luiz C. (Brazil)*** *on 6 Feb 2003, 1:39 p.m.,*
in response to message #1 by Ex-PPC member

Provided the calculator has [RCL][E+] (both HP41 and 42 don't have), where [E+] is the summation key, and [CLE] (Clear Statistics) is executed prior to A and B input (too much rules, I know):

```
[E+]
[RLC][E+]   (case 1 stops here)
[RLC][E+]   (case 2 stops here)
```

With the HP55, [RCL][E+] overwrites both X- and Y-Register contents (maybe others do, I don't know) while others lift the stack one time, leaving previous Y-contents unchanged and moved up to Z-register. In the HP55, we have:

```
[E+]
[ENTER^]
[RLC][E+]   (case 1 stops here)
[ENTER^]
[ENTER^]
[RLC][E+]   (case 2 stops here)
```

To be honest, I didn't like these solutions: needs a lot of memory (all statistical registers must be present)

But it works...

Proof of the superiority of RPL

Message #13 Posted by ***Glen Kilpatrick*** *on 6 Feb 2003, 7:43 p.m.,*
in response to message #1 by Ex-PPC member

Hah! All you need for these is the right "O/S":

[1] OVER SWAP

[2] DUP2

And having poured gasoline on the fire, I'll just step back a few, uhhh, thousand, kilometers :).

Re: Proof of the superiority of RP...N!

Message #14 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 6 Feb 2003, 7:58 p.m.,*
in response to message #13 by Glen Kilpatrick

If you relax the challenge constraints (which are part of the fun), RPN does better than RPL: someone already posted

RCL Y

X<>Y

(two steps)

Within the problem constraints, I am still looking for a better way than:

0) Starting with X Y Z T = a b p q

1) Roll Down produces X Y Z T = b p q a

2) Roll Down produces X Y Z T = p q a b

3) "+" produces X Y Z T = (p+q) a b b

4) Roll Down produces X Y Z T = a b b (p+q)

... q.e.d.

Since this "hole" is Par 3, I am still one over par (4 steps), so this is a Bogey; I asume the (p+q) term in the T register does not violate the constraints.

Note that RPL, among other curses, will not allow for the automatic T register replication used in step 3, and probably will "nag" at you many times while you tinker about this challenge.

3 steps, but with one question....

Message #15 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 6 Feb 2003, 8:30 p.m.,*
in response to message #14 by Andrés C. Rodríguez (Argentina)

I don't have any calculator which has the "RCL E" (RCL sigma) function, but if "RCL E" has the effect of pushing the stack two levels up; then my previous example could be converted in:

0) Start with X Y Z T = a b p q

1) RCL E produces X Y Z T = v w a b,

where v and w are the contents of the x-sum and y-sum registers, which value doesn't matter, the only issue is to push the stack two levels up.

2) "+" produces X Y Z T = (v+w) a b b,

due to T register replication feature of RPN

3) Roll Down produces X Y Z T = a b b (v+w)

... q. e. d. in 3 steps

This will not work in my late HP25, nor in my HP41 or HP42; but perhaps it does in the HP11, 15, etc...

Re: 3 steps, but with one question....

Message #16 Posted by ***Michael F. Coyle*** *on 6 Feb 2003, 8:40 p.m.,*
in response to message #15 by Andrés C. Rodríguez (Argentina)

Hi Andrés,

Good solution! It does work on the 11C.

- Michael

Re: 3 steps, another option

Message #17 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 6 Feb 2003, 8:50 p.m.,*
in response to message #16 by Michael F. Coyle

1) CLR SIGMA (or CLR REGS)

2) E+,

Note that after the Sigma instruction, Y is preserved, X is replaced by a count, which in this case will be 1, and stack lift is NOT enabled...

3) RCL SIGMA

In an HP25, it would take two steps: RCL 4 and RCL 7...

Actually, if cleared registers could be assumed as a starting point, it could be one less step (Birdie)

Re: 3 steps, another option

Message #18 Posted by ***Vieira, Luiz C. (Brazil)*** *on 6 Feb 2003, 9:46 p.m.,*
in response to message #17 by Andrés C. Rodríguez (Argentina)

Hola, Andrés;

as posted by Michael, [f][CLEAR][REG] is better because [f][CLEAR][SIGMA] will clear stack registers' contents in almost all RPN calculators. And both HP41 and HP42 do not have a [RCL][SIGMA+], what is too bad!.

Best regards, my friend. Good reading your posts again.

Re: 3 steps, another option

Message #19 Posted by ***Andres Rodriguez*** *on 7 Feb 2003, 5:16 p.m.,*
in response to message #18 by Vieira, Luiz C. (Brazil)

Luiz: Thank you for your greetings, I have been very busy in the last weeks.

At least in my limited experience, CLEAR SIGMA should not affect the stack contents; in fact, in the very good HP41 emulator I have in my PDA it keeps the stack untouched; later today I will check in my "real" HP41C and HP42S, but I don't have them at hand right now. I cannot think there is a reason for CLEAR SIGMA to clear the stack...

CLEAR SIGMA and the stack

Message #20 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 8 Feb 2003, 9:22 a.m.,*
in response to message #19 by Andres Rodriguez

I have tried a friend's HP27 (Woodstock), my HP41C, my HP42S, my daughter's HP32Sii, and also eV41 on my HP/Compaq WindowsCE-based PDA, and have not found an example of CLEAR SIGMA affecting the stack contents....

Re: CLEAR SIGMA and the stack

Message #21 Posted by ***Vieira, Luiz C. (Brazil)*** *on 8 Feb 2003, 10:22 a.m.,*
in response to message #20 by Andrés C. Rodríguez (Argentina)

Hola, Andrés;

you are correct: CLEAR [SIGMA] clears stack contents of all Voyagers (the worst case is the HP12C where CLEAR REG clears everything but programs) and I also tested in the HP55 (classical) and the stack is cleared with any of both: CLR and CL.R. Both HP67 and 97 do not have an specific CLEAR [SIGMA] and CLREG is used so far. In the HP41 and HP42 it does not happen because they have CLST, not available in any voyager. I do not know about the HP27 and the HP32SII.

What I still wonder about is the LASTx contents: the only one I tested that effectively has LASTx contents cleared after CLEAR REG is the HP12C. I did not test all of them, but all other Voyagers, any HP41, the HP42 and the HP55 keep LASTX contents after any clearing function.

Weird...

Cheers.

Re: CLEAR SIGMA and the stack

Message #22 Posted by ***Sambonator*** *on 8 Feb 2003, 6:05 p.m.,*
in response to message #21 by Vieira, Luiz C. (Brazil)

On the HP 27, clear sigma does not clear the stack. Neither does that function clear the stack on the 32S and 32SII. Sam

Re: CLEAR SIGMA and the stack

Message #23 Posted by ***Vieira, Luiz C. (Brazil)*** *on 8 Feb 2003, 9:22 p.m.,*
in response to message #22 by Sambonator

Hi;

I have a Woodstock HP27 in hands (does not belong to me) I just brought back to life and I looked at the keyboard and saw a [CLEAR][STK] ([f][CLx]), what explains the fact that [CLEAR][SIGMA] keeps stack contents unchanged. Also I remember the HP32S has the ST option at the CLEAR menu; maybe the HP32SII has this option, too.

I believe that, with a few exceptions, calculators that have a [CLEAR][STACK] command/function have the stack contents unchanged when [CLEAR][SIGMA], if available, is performed. As none of the Voyagers have a [CLEAR][STACK] command/function, [CLEAR][SIGMA] also performs a [CLEAR][STACK]. As for the classic models, I can only speak for the HP55, that has the stack contents cleared after CL.R or CLR. In the HP67 manual, p. 80, it's explained that the user should enter 0 and press [ENTER^] three times to clear the entire stack and that [f][CL REG] does not affect its contents.

Wow! That's what I call a thread to exhaust a particular theme.

Cheers.

Re: 3 steps, but with one question....

Message #24 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 6 Feb 2003, 8:52 p.m.,*
in response to message #16 by Michael F. Coyle

Thank you, I liked your solution to #2

Re: RPN challenge -- Not quite 3 steps, but a solution to the general problem . . .

Message #25 Posted by ***Paul Brogger*** *on 7 Feb 2003, 3:23 p.m.,*
in response to message #1 by Ex-PPC member

Maybe someone can program a solution to the **general problem** in fewer steps?

Given any values in *x, y, z, t* & *last x*, return the four stack registers with any of those values present in ***any*** **arbitrary sequence** of selection and/or repetition, as specified by the user (or with slight adaptation, by another program).

```
    A01  LBL A        "XEQ A" to start "The Scrambler"


    A02  STO B        x will be referred to by "1"
    A03  Roll Down      access y
    A04  STO C        y will be referred to by "2"
    A05  Roll Down      access z
    A06  STO D        z will be referred to by "3"
    A07  Roll Down      access t
    A08  STO E        t will be referred to by "4"
```

```
A09  last x          access last x
A10  STO F         last x will be referred to by "5"
A11  STO G                    . . . or "6"
A12  STO H                      . . . or "7"
A13  STO I                        . . . or "8"
A14  STO J                          . . . or "9".


A15  0             access zero
A16  STO A           zero will be indexed as "0"


A17  INPUT V       input the desired sequence vector
A18  RCL V         first,
A19  ABS              insure not negative
A20  IP                  and no fractional part
A21  STO V                  in vector.


A22  XEQ V         obtain specified t value
A23  XEQ V         obtain specified z value
A24  XEQ V         obtain specified y value
A25  XEQ V         obtain specified x value


A26  RTN


V01  LBL  V        use low-order digit of V to obtain next register


V02  10            need ten to obtain a digit
V03  STO / V       shift V one digit right
V04  STO  i          and preset index register
V05  ROLL DOWN         keep stack from overflowing


V06  RCL  V        obtain shifted vector
V07  FP              reduce to newly-obtained digit
V08  STO - V            and remove that from V
V09  STO * i               while restoring it as an integer
V10  ROLL DOWN               prevent stack overflow


V11  1              need to increment index by 1
V12  STO + i          to keep from (unfortunate) error
V13  ROLL DOWN          prevent stack overflow
```

```
   V14  RCL (i)       recover next value desired


   V15  RTN
```

With any values in the stack & *last x*, XEQ A.
It'll prompt for a “V” value.
Enter four digits in *x, y, z, t* sequence, specifying any desired choice of register values using the following encoding:

```
  0 == zero (cleared)
  1 == original x value
  2 == original y value
  3 == original z value
  4 == original t value
5-9 == original last x value
```

Whatever is entered (I hope!), the low-order four digits of the integer portion are interpreted as a specification of which values are wanted where, and the stack is returned in the condition so indicated.

This was programmed on an HP-32SII, and (if I’ve transcribed everything correctly) “MEM” shows LBL A = 39.0 bytes, LBL V = 22.5 .

With a few demands added regarding proper user behavior, some of the normalization and error prevention lines could be removed . Deleting lines A11-14 and A18-21 reduces the LBL A footprint to 27.0 bytes. (This, then, requires the user to enter a positive integer, and to use only codes “0”-“5”.)

To make it work as a subroutine (who on *earth* would bother?), simply delete line A17 and XEQ A from elsewhere.

And . . .

with regard to the two special cases posed by **Ex-PPC member**, one would XEQ A and enter “122n” (where “n” is any decimal digit) to achieve the first, or “1212” to satisfy the second.

-- "It is thus", cried Alexander, "that I untie *all* Gordian knots!"

Roll-Stack Population Register _aka_ Stack-only Solution in Three Steps

Message #26 Posted by ***Glen Kilpatrick*** *on 8 Feb 2003, 11:04 a.m.,*
in response to message #1 by Ex-PPC member

I accidentally discovered this last night, present it for mutual amusement. I don't consider it a "real" solution to the presented problem, but instead more of a disclosure of an incompletely presented RPN (by HP, no less! :).

I recently got a 17Bii from Randy Sloyer (THANKS!), and it differs from a 17B primarily in having RPN (so they say, but read on). So carefully follow the bread crumbs below to See What Happens:

[A] Shift MODES RPN ---> see RPN MODE

[B] Shift CLEAR DATA ---> 0.00

[C] ((((---> 0.00 repeated, just like you'd expect; note that in RPN mode, "(" is the Stack Roll Down.

[D] 4 INPUT 3 INPUT 2 INPUT 1 ((((---> yept, Life Is Good, see 1.00, 2.00, 3.00, & 4.00

[E] v & ^ ---> Roll Down & Roll Up work just like we're familiar with; note these are the all-purpose roll keys between the Shift and the INPUT keys.

[F] Shift CLEAR DATA 1 v v v ---> hmmm, all we see are 1.00, this doesn't seem right.

[G] Shift CLEAR DATA 2 INPUT 1 v v v v v ---> alternating 2.00 & 1.00, _no_ zeros.

[H] Shift CLEAR DATA 3 INPUT 2 INPUT 1 v v v v v ---> 2.00, 3.00, 1.00, 2.00, 3.00 ..., and this is beginning to make sense (^ gives you the opposite sense, confirms the stack roll goes up too).

So [F] through [H] show you that the stack is not a constant four levels TZYX, but instead starts out at one level, and is "populated" as appropriate.

[I] Shift CLEAR DATA) 1 v v v v v ---> 0.00, 0.00, 1.00, 0.00, 0.00, 1.00 ...; ")" or X<>Y seems to be the exception, SWAP populates levels 1 & 2 (RPL terminology seems more appropriate now), and a numeric entry pushes them up to 2 & 3.

Having now presented the rationale (and examples for those not endowed with a 17Bii), I present a three-step, stack-only solution to the original problem [1] B A ---> B B A:

Shift CLEAR DATA "B" INPUT "A" ---> Setup of initial conditions; we now have a two-stack, B over A.

^ or v or (---> we now have a two-stack, A over B

INPUT ---> three-stack, A over B over B

^ ---> voila!, B over B over A (and indeed, we don't care about the "T" register, it's not yet been "invented".

[J] + + + + + + v v v v v ^ ^ ^ ^ ---> 8.00 repeated (remember, we're coming from the Z:3 Y:3 X:2 of above, and I use those "levels" loosely); those two additions appear to "consume" levels 3 & 2, so that we're left only with level 1 (not exactly RPL behavior, the third "+" would error with it).

This experiment appears to suggest an alternate and cleaner view, that we really are adding repeatedly "0.00" to X, and that we really have an empty T-register (and four stack levels, whew, we're back to RPN). However, there's a "Roll-Stack Population Register" somewhere that advises the roll keys how high to roll. On a true RPN this would always be "four", but on a 17Bii it apparently can be anything between one and four.

[K] 4 INPUT 3 INPUT 2 INPUT 1 + + + + + + + + + ---> 3.00, 6.00, 10.00, 14.00, 18.00, 22.00 ...; now *this* is our familiar T-register (RPN) behavior.

Conclusion: Until you have all four levels populated (that is, your Roll-Stack Population Register is four), don't trust your 17Bii to Roll in true RPN fashion.

Perhaps somebody with a 19Bii can advise how faithful its RPN behavior is.

P.S. My hat's off to Paul Brogger for his "Re: RPN challenge -- Not quite 3 steps, but a solution to the general problem . . . ", most amazing. As I don't have a true RPN (and 17Bii's don't do procedural programming anyway), I can't test it, only admire it.

Re: Roll-Stack Population Register _aka_ Stack-only Solution in Three Steps

Message #27 Posted by ***Vieira, Luiz C. (Brazil)*** *on 8 Feb 2003, 1:41 p.m.,*
in response to message #26 by Glen Kilpatrick

Hi, Glen;

Yes, the HP17BII (and the HP19BII as well) have a 1-to-4 level stack. It may contain from one to four numbers stored on it,and when you clear it, it will contain only one register - the X-register - with a [0.00]. Try this:

```
[][CLEAR DATA]          ([] is the shift key)
0 [INPUT][INPUT][INPUT] 4 (any number)
[(][(][(][(]
```

What do you see? You now have the four stack-registers. If you simply clear data and enter 4 (any number), there will be only the X-register and the number 4.00 stored in there.

If you enter only three numbers, [(] (or roll-down) will roll only these three registers. And it happens the same in the HP19BII.

There is a small book that came with my HP19BII only to explain RPN and the differences between the HP19B and the HP19BII. I can scan it (it's just a few pages) but I must tell you it's written in Spanish.

Cheers.