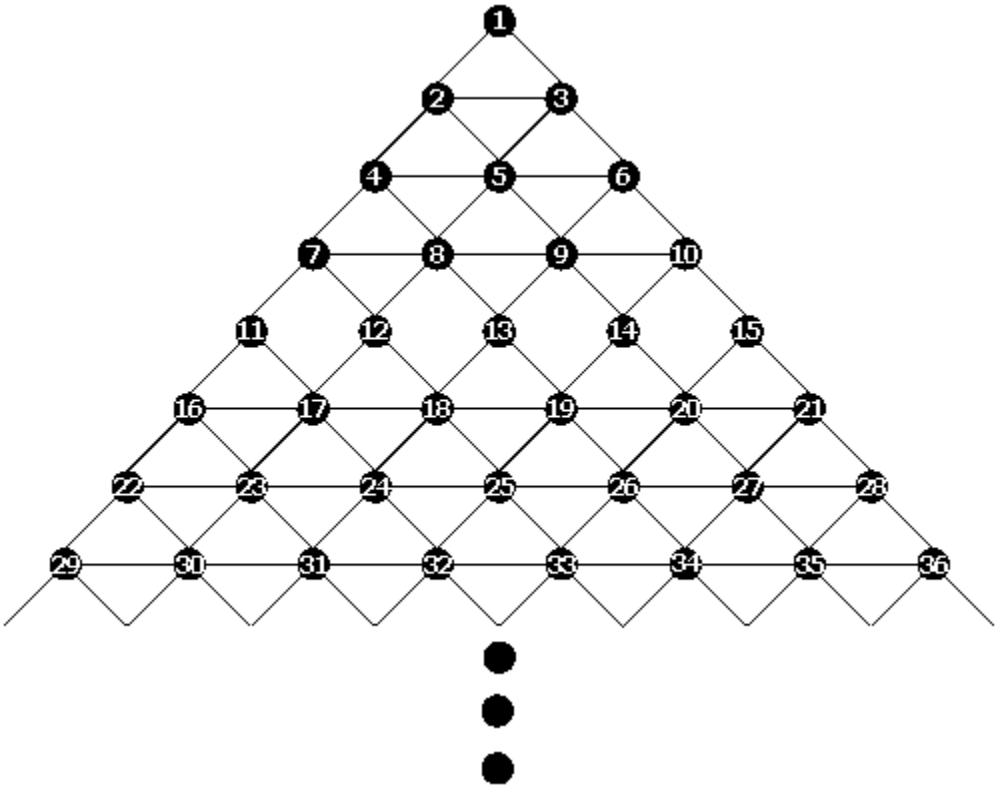
Appendix D - HHC 2008 Programming Contest

Gene Wright

Consider the points on a grid of equilateral triangles as shown below. Note that if the points are numbered from left to right and top to bottom, then groups of these points form the vertices of certain geometric shapes. For example, the sets of points 1, 2 ,3 and 7, 9, 18 are the vertices of triangles, the sets 11,13,24,26 and 2,7,9,18 are the vertices of parallelograms, and the sets 4, 5, 9, 13, 12, 7 and 8, 10, 21, 34, 32, 17 are the vertices of hexagons.



Write a program named A which will accept a set of points on this triangular grid, analyze it and determine whether the points are vertices of a triangle, parallelogram, hexagon or an illegal figure. In order for a figure to be acceptable, it must meet two conditions:

1) Each side of the figure must coincide with an edge in the grid, and

2) All sides of the figure must be of the same length.

INPUT: The input will consist of a series of point sets. Each point set will have at most six points in a set. The points in the set are limited to the range of 1 through 105.

35s (or other RPN model): Input will be done one point at a time. Each point will be keyed and R/S will be pressed. A -1 will be entered when all data points are entered and R/S will be pressed. The input for the data set { 1 2 3 } would be SHIFT CLEAR VARS then 1 STO A 2 STO B 3 STO C 1 CHS STO D. Variables A through G might be used in this manner. The last register used (in order) would contain the -1 value. Running the program will be done after storing the inputs by XEQ A ENTER.

50g (or other RPL model): The stack will be clear except for a list containing the data points. The input for the sample data set { 1 2 3 } would be { 1 2 3 } placed on a clear stack. Running the program will be done by pressing: VAR then the menu label A. USER RPL only. No unsupported entry points, System RPL, etc.

OUTPUT:

35s (or other RPN model): Display a 0 if the set is an invalid figure. Display a 1 if the set is a triangle. Display a 2 if the set is a parallelogram. Display a 3 if the set is a hexagon.

50g (or other RPL model): Display "ERROR", "TRIANGLE", "PARALLELOGRAM", or "HEXAGON", appropriately.

TEST: Fastest total time to evaluate a set of input test cases. Decision of the judge is ABSOLUTELY final.

==========

The contest generated only a handful of entries, perhaps because of the apparent complexity of the problem or perhaps because of the full schedule of the conference.

There were two classes of machines eligible: RPL (any) and RPN (any). Only four entries were received, two in each category.

Only one machine (a 50g) correctly solved all input problems. That was the winner of course. Allen Thomson. He correctly solved all 7 input cases in about 12 seconds. Timing wasn't as critical, since his program was the only one that worked in all cases.

One note: The hexagons in the problem are to be regular hexagons - no internal pointing sides.

Perhaps most interestingly, when this problem was posted to the HP Museum forum after the contest, quite a bit of interest ensued. Several of these were posted to the forum at this link:

http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv018.cgi?read=141243

A couple of the more interesting programs generated there are reproduced below. Both of these are by Egan Ford.

Fast RPL solution #1

```
HHC 2008 Programming Contest (and a tip for future UserRPL optimization)
Message #40 Posted on the HP Museum websit3 by Egan Ford on 12 Oct 2008, 12:43 p.m.,

I got the time down to 1.25 seconds for all 7 problems. An incredible 2x+ increase in
performance. And that is Gjermund's UserRPL tip: Floats are faster than Ints.

Thanks Gjermund!

The changes:

   1. Added a . (dot) after each integer.
   2. Converted the input list to floats (I->R after SORT).

%%HP: T(3)A(R)F(.);
\<< SORT I\->R \-> p
  \<< "ERROR" p SIZE
    CASE DUP 3. ==
      THEN DROP p OBJ\-> \-> a b c r
        \<<
          b 8. * 1. + \v/ 1. - 2. / CEIL 'r' STO
          c b - b a - >
          IF
          THEN
```

```
      a r r 1. - * 2. / >
      IF
      THEN
        b a - DUP r 2. * + 1. - * 2. / b + c ==
        IF
        THEN DROP "TRIANGLE DOWN"
        END
      END
    ELSE
      c r r 1. + * 2. / \<=
      IF
      THEN
        b c b - DUP NEG r 2. * + 1. - * 2. / - a ==
        IF
        THEN DROP "TRIANGLE UP"
        END
      END
    END
  \>>
END DUP 4 ==
THEN DROP p OBJ\-> \-> a b c d r
  \<<
    c 8. * 1. + \v/ 1. - 2. / CEIL 'r' STO
    b r r 1. - * 2. / >
    IF
    THEN
      c b - DUP r 2. * + 1. - * 2. / c + d ==
      IF
      THEN b c b - DUP NEG r 2. * + 1. - * 2. / - a ==
        IF
        THEN DROP "PARALLELOGRAM DIAMOND"
        END
      END
    ELSE d c - b a - ==
      IF
      THEN d r r 1. + * 2. / \<=
        IF
        THEN
          b 8. * 1. + \v/ 1. - 2. / CEIL 'r' STO
          a r r 1. - * 2. / >
          IF
          THEN
            b a - DUP r 2. * + 1. - * 2. / b + d ==
            IF
            THEN DROP "PARALLELOGRAM LEFT"
            ELSE
              b a - DUP r 1. + 2. * + 1. - * 2. / b + d ==
              IF
              THEN DROP "PARALLELOGRAM RIGHT"
              END
            END
          END
        END
      END
    END
  \>>
END 6 ==
THEN p OBJ\-> \-> a b c d e f r
  \<< f e - b a - ==
    IF
    THEN f e - 2. * d c - ==
      IF
      THEN
        f 8. * 1. + \v/ 1. - 2. / CEIL 'r' STO
```

```
            e r r 1. - * 2. / >
            IF
            THEN
              b 8. * 1. + \v/ 1. - 2. / CEIL 'r' STO
              a r r 1. - * 2. / >
              IF
              THEN
                b a - DUP r 2. * + 1. - * 2. / a + c ==
                IF
                THEN
                  r b a - + 'r' STO
                  b a - DUP r 2. * + 1. - * 2. / d + f ==
                  IF
                  THEN DROP "HEXAGON"
                  END
                END
              END
            END
          END
        END
      \>>
    END
  END
\>>
\>>
```

Fast RPN solution. About 44 seconds on an HP 41CX.
HHC 2008 Programming Contest -- anyone want to try an RPN solution? Message #22 Posted on HP Museum website by Egan Ford on 4 Oct 2008, 3:55 a.m.,

```
01 LBL "SETUP"        134 *                 267 1                 400 RCL 02
02 0                  135 2                 268 +                 401 8
03 SETSW              136 /                 269 SQRT              402 *
04 CLRG               137 RCL 01            270 1                 403 1
05 11                 138 X<=Y?             271 -                 404 +
06 STO 09             139 GTO 20            272 2                 405 SQRT
07 0                  140 RCL 02            273 /                 406 1
08 RTN                141 RCL 01            274 STO 07            407 -
09 LBL "FS"           142 -                 275 FRC               408 2
10 FIX 00             143 ENTER             276 X=0?              409 /
11 CF 29              144 ENTER             277 GTO 09            410 STO 07
12 CLA                145 RCL 07            278 RCL 07            411 FRC
13 >"NUM PTS?"        146 2                 279 1                 412 X=0?
14 PROMPT             147 *                 280 +                 413 GTO 13
15 INT                148 +                 281 INT               414 RCL 07
16 STO 00             149 1                 282 STO 07            415 1
17 1000               150 -                 283 LBL 09            416 +
18 /                  151 *                 284 RCL 07            417 INT
19 1                  152 2                 285 1                 418 STO 07
20 +                  153 /                 286 -                 419 LBL 13
21 STO 10             154 RCL 02            287 RCL 07            420 RCL 07
22 LBL 00             155 +                 288 *                 421 1
23 CLA                156 RCL 03            289 2                 422 -
24 >"PT "             157 X#Y?              290 /                 423 RCL 07
25 ARCL 10            158 GTO 20            291 RCL 01            424 *
26 >"?"               159 CLA               292 X<=Y?             425 2
27 PROMPT             160 >"TRIANGLE DOWN”  293 GTO 20            426 /
28 INT                161 2                 294 RCL 02            427 RCL 01
29 STO IND 10         162 STO 08            295 RCL 01            428 X<=Y?
30 ISG 10             163 GTO 20            296 -                 429 GTO 20
31 GTO 00             164 LBL 06            297 ENTER             430 RCL 06
32 RUNSW              165 0                 298 ENTER             431 RCL 05
33 FIX 04             166 4                 299 RCL 07            432 -
34 RCL 00             167 X#NN?             300 2                 433 ENTER
```

```
35 1000
36 /
37 1
38 +
39 SIGN
40 LBL 01
41 LASTX
42 LASTX
43 RCL IND L
44 LBL 02
45 X<=NN?
46 GTO 03
47 X<>Y
48 STO Y
49 RCL IND X
50 LBL 03
51 ISG Y
52 GTO 02
53 X<> IND L
54 STO IND Z
55 ISG L
56 GTO 01
57 CLA
58 >"ERROR"
59 0
60 STO 08
61 0
62 3
63 X#NN?
64 GTO 06
65 RCL 02
66 8
67 *
68 1
69 +
70 SQRT
71 1
72 -
73 2
74 /
75 STO 07
76 FRC
77 X=0?
78 GTO 04
79 RCL 07
80 1
81 +
82 INT
83 STO 07
84 LBL 04
85 RCL 02
86 RCL 01
87 -
88 RCL 03
89 RCL 02
90 -
91 X>Y?
92 GTO 05
93 RCL 07
94 1
95 +
96 RCL 07
97 *
98 2
168 GTO 11
169 RCL 03
170 8
171 *
172 1
173 +
174 SQRT
175 1
176 -
177 2
178 /
179 STO 07
180 FRC
181 X=0?
182 GTO 07
183 RCL 07
184 1
185 +
186 INT
187 STO 07
188 LBL 07
189 RCL 07
190 1
191 -
192 RCL 07
193 *
194 2
195 /
196 RCL 02
197 X<=Y?
198 GTO 08
199 RCL 03
200 RCL 02
201 -
202 ENTER
203 ENTER
204 RCL 07
205 2
206 *
207 +
208 1
209 -
210 *
211 2
212 /
213 RCL 03
214 +
215 RCL 04
216 X#Y?
217 GTO 20
218 RCL 03
219 RCL 02
220 -
221 ENTER
222 ENTER
223 CHS
224 RCL 07
225 2
226 *
227 +
228 1
229 -
230 *
231 2
301 *
302 +
303 1
304 -
305 *
306 2
307 /
308 RCL 02
309 +
310 RCL 04
311 X#Y?
312 GTO 10
313 CLA
314 >"PARALLELOGRAM"
315 >" LEFT"
316 4
317 STO 08
318 GTO 20
319 LBL 10
320 1
321 ST+ 07
322 RCL 02
323 RCL 01
324 -
325 ENTER
326 ENTER
327 RCL 07
328 2
329 *
330 +
331 1
332 -
333 *
334 2
335 /
336 RCL 02
337 +
338 RCL 04
339 X#Y?
340 GTO 20
341 CLA
342 >"PARALLELOGRAM"
343 >" RIGHT"
344 5
345 STO 08
346 GTO 20
347 LBL 11
348 0
349 6
350 X#NN?
351 GTO 20
352 RCL 06
353 RCL 05
354 -
355 RCL 02
356 RCL 01
357 -
358 X#Y?
359 GTO 20
360 RCL 06
361 RCL 05
362 -
363 2
364 *
434 ENTER
435 RCL 07
436 2
437 *
438 +
439 1
440 -
441 *
442 2
443 /
444 RCL 01
445 +
446 RCL 03
447 X#Y?
448 GTO 20
449 RCL 06
450 RCL 05
451 -
452 ST+ 07
453 ENTER
454 ENTER
455 RCL 07
456 2
457 *
458 +
459 1
460 -
461 *
462 2
463 /
464 RCL 04
465 +
466 RCL 06
467 X#Y?
468 GTO 20
469 CLA
470 >"HEXAGON"
471 6
472 STO 08
473 LBL 20
474 STOPSW
475 RCL 08
476 STO IND 09
477 1
478 ST+ 09
479 AVIEW
480 RTN
481 LBL "PRINT"
482 RCL 09
483 1
484 -
485 1000
486 /
487 11
488 +
489 STO 10
490 LBL 21
491 CLA
492 RCL IND 10
493 30
494 +
495 INT
496 GTO IND X
497 LBL 30
```

```
99 /
100 RCL 03
101 X>Y?
102 GTO 20
103 RCL 03
104 RCL 02
105 -
106 ENTER
107 ENTER
108 CHS
109 RCL 07
110 2
111 *
112 +
113 1
114 -
115 *
116 2
117 /
118 RCL 02
119 X<>Y
120 -
121 RCL 01
122 X#Y?
123 GTO 20
124 CLA
125 >"TRIANGLE UP"
126 1
127 STO 08
128 GTO 20
129 LBL 05
130 RCL 07
131 1
132 -
133 RCL 07
232 /
233 RCL 02
234 X<>Y
235 -
236 RCL 01
237 X#Y?
238 GTO 20
239 CLA
240 >"PARALLELOGRAM"
241 >" DIAMOND"
242 3
243 STO 08
244 GTO 20
245 LBL 08
246 RCL 04
247 RCL 03
248 -
249 RCL 02
250 RCL 01
251 -
252 X#Y?
253 GTO 20
254 RCL 07
255 1
256 +
257 RCL 07
258 *
259 2
260 /
261 RCL 04
262 X>Y?
263 GTO 20
264 RCL 02
265 8
266 *
365 RCL 04
366 RCL 03
367 -
368 X#Y?
369 GTO 20
370 RCL 06
371 8
372 *
373 1
374 +
375 SQRT
376 1
377 -
378 2
379 /
380 STO 07
381 FRC
382 X=0?
383 GTO 12
384 RCL 07
385 1
386 +
387 INT
388 STO 07
389 LBL 12
390 RCL 07
391 1
392 -
393 RCL 07
394 *
395 2
396 /
397 RCL 05
398 X<=Y?
399 GTO 20
498 >"ERROR"
499 GTO 40
500 LBL 31
501 >"TRIANGLE UP"
502 GTO 40
503 LBL 32
504 >"TRIANGLE DOWN"
505 GTO 40
506 LBL 33
507 >"PARALLELOGRAM"
508 >" DIAMOND"
509 GTO 40
510 LBL 34
511 >"PARALLELOGRAM"
512 >" LEFT"
513 GTO 40
514 LBL 35
515 >"PARALLELOGRAM"
516 >" RIGHT"
517 GTO 40
518 LBL 36
519 >"HEXAGON"
520 LBL 40
521 AVIEW
522 ISG 10
523 GTO 21
524 FIX 06
525 RCLSW
526 CLA
527 >"TIME: "
528 ATIME24
529 AVIEW
530 RTN
531 END
```

HHC 2009 Programming Challenge

Finding partial sums of rows of Pascal's Triangle

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
  etc
```

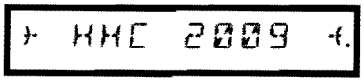
Write an HP50g program which takes a positive integer **n** of any size in level 2 and any integer **r** in level 1 and returns the sum of C(n,0) + C(n,1) + ... + C(n,r) to level 1. The output should be 0 for **r** < 0 and $2^{\mathbf{n}}$ for **r** ≥**n**.

The program will be transferred from your HP50g to the judges' HP50g and will be tested (in exact mode) for a selection of values of **n** and **r** chosen by the judges. The fastest program wins.

The usual rules apply:

- The program must be in user code only
- The program must be self-contained
- The program must leave the stack unchanged except for replacing **n** and **r** with the result.
- Default flag settings are assumed, except that RPN mode must be set. Flag settings must be restored if changed.
- The judges' decision is final.

Have fun!



HP Calculator Programming Contest

Sept 25-26, 2010 / Fort Collins, Colorado

Goal

HP 15C or 15c+: Write a program that sorts registers 1-9 into ascending order.

RPL: Write a program that takes any size list containing only integers, and sorts the list into ascending order with the odd numbers first followed by the even numbers.

Specifics

HP-15C or 15c+: The judges will place values into registers 1-9. These will be valid, ordinary integer numbers. There is no output per se; the sorted values are to be left in registers 1-9. The winner will be the program with the fewest program steps.

RPL: The judges will place a list of values onto level 1 of the stack. These will be valid, ordinary integer numbers. The output is the sorted list in stack level 1 with the odd numbers first followed by the even numbers. The winner will be the program with the smallest byte count multiplied by the runtime over a run of 4 sample inputs.

Rules

1. The decision of the judges is final.
2. The purpose of the contest is to have fun.
3. At least two contestants must participate.
4. Plain-vanilla user code only; no synthetics or SYSEVALs or other monkey business.
5. Contest program submittal must be completed by the end of the contest (TBA).
6. **HP-15C or 15c+:** Take your calculator pre-loaded with your program to the judges.

RPL: Include your initials in the name of your program and transfer your program to the judges' machine.

7. This is a contest between *individuals, not teams*; one submittal per person, one person per submittal.
8. By submitting a program, you agree to allow it to be shared with the community.
9. You must to be present to win.
10. If a point is unclear, ***ask*** immediately. No excuses for ignorance. Clarifications will be officially announced during conference hours.
11. Assume machine default flag settings (except HP49/50; assume RPL mode). Altered flag settings must be returned to default status upon program completion.
12. *Work alone*. Do not consult the Internet. The AUR, Owner's Manual, or other books are permissible, of course.
13. In case of a tie, the most elegant solution (according to the judges) wins. In case of identical winning programs, the first one submitted wins. The judges' decision is final.
14. **H**appy **P**rogramming! *-jkh-*

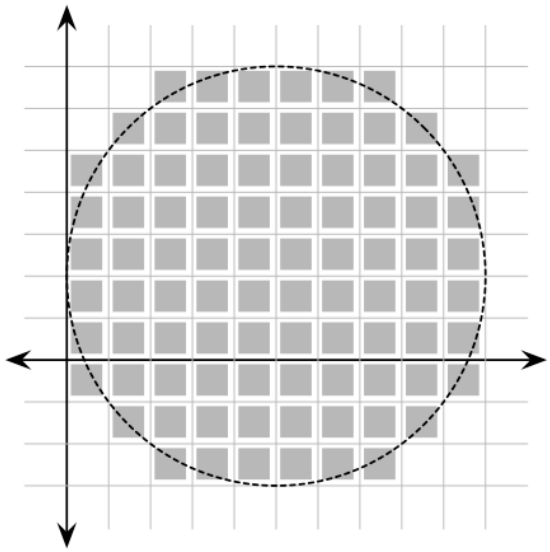
HP Calculator Programming Contest

Sept 25-26, 2010 / Fort Collins, Colorado

HHC 2011 RPN Programming Contest

September 24-25 / 2011 San Diego

Problem Description: Did you know that if you draw a circle that fills the screen on your 1080p high definition display, almost a million pixels are lit? That's a lot of pixels! But do you know exactly how many pixels are lit? Let's find out!



Assume the display is set on a Cartesian grid where every pixel is a perfect unit square. For example, one pixel occupies the area of a square with corners (0,0) and (1,1). A circle can be drawn by specifying its center in grid coordinates and its radius. A pixel on the display is lit if any part of is covered by the circle; pixels whose edge or corner are just touched by the circle, however, are not lit.

You must compute the exact number of pixels "lit" when a circle with a given position and radius is drawn.

Input: Each test case consists of three integers, x, y, and r ($1 \le x, y, r \le 5000$), specifying respectively the center (x, y) and radius of the circle drawn. The radius will be loaded into stack register Z, the y coordinate of the center of the circle into stack register Y, and the x coordinate of the circle into stack register X. Assume successive program runs are to be started by simply entering new values and pressing R/S. Assume that all circles fit on the display panel even if in reality they would not.

Output: Return the number of pixels that are lit when the specified circle is drawn.

Sample Cases: (A) Input of 1 ENTER 1 ENTER 1 R/S should return 4. This represents a circle with a center of (1,1) and a radius of 1. The display would have 4 pixels "on" to represent this circle. (B) Input of 5 ENTER 2 ENTER 5 R/S should return 88. This represents a circle with a center of (5,2) and a radius of 5. The display would have 88 pixels "on" to represent this circle. This is the circle shown in the figure above. 88 pixels are "on" in this picture.

Machines Eligible: This contest is open to any and all RPN machines: 15c, 15c+, 15c LE, 34S, 41CL, 42S, 67, 65, etc. RPL users are welcome to try the problem, but this is for RPN machines only.

Rules: (aka the fine print)

1) The decision of the judge is FINAL. No appeals are allowed to anyone or anything.
2) The purpose of this contest is to have fun and learn.
3) At least two contestants must submit an entry.
4) No custom built ROM or machine code can be built and used for this problem. Any already existing functionality in the machine is ok.
5) You must submit a machine with your program already keyed in to the judge AS WELL as a legible listing of your program with your name on the listing AND the machine. Machines with no names that are given to the judge are assumed to be gifts to the judge. Thank you!
6) Submission must be made by the end of the contest (Time is TBA).
7) Assume the program will start running with step 001 and/or a R/S.
8) By submitting a program, you agree to allow it to be shared with the community.
9) This is a contest between individuals, not teams. One submittal <> one person.
10) You may not access the internet for any help in any fashion. Do not cheat in any way. Do not check the HP Museum Forum either.
11) You must be present to win.
12) If a point is unclear, ask immediately. No excuses for ignorance. Clarifications will be shared with the entire group during the conference.
13) Assume default machine settings. Your program must stop with the default settings in place.
14) Winner will be the program with the fastest times over the test cases giving correct results. If in the judge's sole discretion, two entries are "about the same speed," the winner will be the shortest routine. In case of a tie, the most elegant solution (according to the judge) wins.
15) The purpose of this contest is to learn and have fun. Happy Programming.

HHC 2011 Programming Contest

The Problem

Write a program for the HP50g in RPN Mode which takes a non-empty string of any length consisting of some or all of the 26 letters A, B, C … Z and returns, as a type 28 integer, the exact number of distinct arrangements of these letters in the string. (Permuting multiple occurrences of the same letter does not change the arrangement.)

Examples

"DEEDED"	→	20
"ANTITRINITARIAN"	→	126126000
"ABCDEFGHIJKLMNOPQRST"	→	2432902008176640000
“AAAAABBBBBCCCCCDDDDDEEEEE"	→	623360743125120

Note that these results are 6!/3!3!, 15!/3!3!3!4!2!, 20!/1!^20 and 25!/5!^5 respectively.

The Rules

1. The program must be a (self-contained) single object in user code which does not call it self by name.
2. Default flag settings (except for flag –95) are assumed and must be restored if changed.
3. The stack, apart from input and output, must be left as found.
4. The program must not contain KILL or otherwise interfere with the programmatic testing and evaluation of submissions.
5. Your program must be transferred to the judge’s machine under some identifying three-letter name before the announced deadline.
6. The winning program will be the one for which size*speed (bytes*sec) is least, where the speed of execution will be determined for one or more longer input strings, probably of several hundred letters, chosen by the judge.
7. The purpose of the contest is to have fun and the decision of the judge is final.

HHC 2011

Results of the HHC 2011 Programming Contest for the HP 50g

Bill Butler

The winner was Roger Hill and the runner-up Jacob Wall.

The announced deadline was by lunch Sunday and submissions were received and judged by Bill Butler (who, in turn, is submitting this report).

There were six entries – in order of submission ghs, DMH, JW4, ELS, CDB and RCH. Four were determined to work as required. The remaining two were ELS which gave results as type 0. integers (and hence rounded-off rather than exact answers) and ghs which left the input on the stack contrary to (at least the intention of) the instructions although it could be argued that these instructions were not sufficiently explicit about this. Fortunately for the judging the program in question would not otherwise have won. Corrections for these two programs involve but one extra command each – for ELS insert R−−>I between the first SIZE and FACT and for ghs append NIP at the end. The resulting amended programs ES2 and gs2 are listed below and will be included in the comparisons along with two additional programs CB2 and CB3 submitted by Cyrille de Brebisson after the deadline, but before the results were announced (to see how they would have fared), understanding that they were not part of the competition. Also included are four test programs of mine from before the contest, written of course with no pressure, adequate sleep and unlimited time. All twelve programs pass my initial test run with eight carefully chosen inputs including the four examples given. The timing itself was for strings consisting of repetitions of A through Z cut off at 200, 400 and 600 characters respectively. (The outputs are integers of 263, 542 and 823 digits respectively.) Run times (sec) and scores (bytes*sec) are given for these twelve programs and three inputs where it is understood that only the four first-listed programs were actually in the running. It is of interest that these four programs ranked in order of decreasing size. Available memory for the timing was around 200K although memory is not particularly at issue here.

			200 chars		400 chars		600 chars	
RCH	6E69h	142.	4.095	581.4	9.140	1297.9	15.347	2179.3
JW4	555h	141.	4.590	647.2	11.689	1648.1	23.279	3282.4
CDB	471h	135.	4.507	608.5	13.510	1823.8	26.179	3534.1
DMH	4B5Dh	127.	12.839	1630.6	33.453	4248.5	61.103	7760.1
ES2	980Bh	124.	8.168	1012.9	19.348	2399.2	34.888	4326.1
gs2	6378h	136.	9.029	1228.0	20.961	2850.7	37.573	5110.0
CB2	522Bh	98.5	2.240	220.6	8.404	827.8	18.191	1791.8
CB3	D71Ch	78.	2.169	169.2	8.332	649.9	18.120	1413.4
XX1	8566h	63.	2.162	136.2	8.327	524.6	18.106	1140.7
XX2	4F57h	74.5	2.074	154.5	4.881	363.6	8.540	636.3
XX3	1D0Fh	80.5	2.028	163.2	4.707	378.9	8.233	662.7
XX4	1564h	80.5	2.041	164.3	4.722	380.1	8.345	671.8

Listings for the twelve programs follow. All involve interesting ideas and techniques and one can learn from them all. The character "■" in Roger's program is character 26. (although simply using the integer 26 at 6.5 bytes for "■" NUM would have fared better). The empty strings "" in programs XX1 through XX4 have been entered as counted strings C$ 0 at a saving of 2.5 bytes although it is noted that they would have to be so re-entered whenever the programs are edited.

```
RCH  #6E69h  142.
<< 0.  DUP  5.  SQ  NDUPN  2. +
   ROLL  DUP  SIZE  DUP  R−−>I  UNROT  1  SWAP
   START  DUP  NUM  61.  −  DUP  PICK
         1.  +  SWAP  1.  −  UNPICK  TAIL
   NEXT  DROP  1  1.  "■"  NUM
   START  UNROT  SWAP  R−−>I  DUP2  −
        UNROT  COMB  ROT  *
   NEXT  + >>

JW4  #555h  141.
<< { }  1  PICK3  SIZE  DUP  R−−>I  !  5  ROLLD
   FOR  i  OVER  i DUP  SUB  DUP2  POS
        IF  THEN  DROP  ELSE  +  END
    NEXT  1  DUP  PICK3  SIZE
    FOR  i  ROT  PICK3  i  GET  ""  SREPL
         R−−>I  !  ROT  *  ROT  SWAP
    NEXT  NIP  NIP  / >>

CDB  #471h  135.
<< 0  26.  NDUPN  1.  +  ROLL  1.  OVER  SIZE
   FOR  A  DUP  A  DUP  SUB  NUM  62.  −  DUP
        ROLL  1  +  SWAP  1.  −  ROLLD
   NEXT  SIZE  R−−>I  !  1.  26.
   FOR  A  SWAP  !  /  NEXT >>

DMH  #4B5Dh  127.
<< 1  SWAP  0.  26.  NDUPN  −−>ARRY  OVER  SIZE  1.  SWAP
   FOR  I  OVER  I  DUP  SUB  NUM  64.  −  DUP2  GET  1.  +
        PUT  LASTARG  6.  ROLL  I  R−−>I  *  SWAP  R−−>I  /
        NIP  NIP  UNROT
   NEXT  DROP2 >>

ES2  #980Bh  124.
<< −−>  S
   <<  S  SIZE  R−−>I  FACT  { 0 }  1  5
       START  DUP  +  NEXT  1  S  SIZE
       FOR  I  S  I  DUP  SUB  NUM  64  −
            DUP2  GET  1  +  PUT
       NEXT  FACT  ΠLIST  /  >> >>


gs2  #6378h  136.
<< DUP  { }  1  26  FOR  K  0 +  NEXT
   OVER  SIZE  R−−>I  UNROT  PICK3  1  SWAP
   FOR  K  OVER  HEAD  NUM  64  −  DUP2
        GET  1  +  PUT  SWAP  TAIL  SWAP
   NEXT  FACT  ΠLIST  SWAP  DROP
   SWAP  FACT  SWAP  /  NIP >>
```

```
CB2  #522Bh  98.5
<< DUP  SIZE  R−−>I  !  SWAP  64.
    WHILE  1.  +  OVER  SIZE
    REPEAT  SWAP  OVER  CHR  ""  SREPL  R−−>I  !
          4.  ROLL  SWAP  /  ROT  ROT  SWAP
    END  DROP2 >>

CB3  #D71Ch  78.
<< DUP  SIZE  R−−>I  !  SWAP
    WHILE  DUP  SIZE
    REPEAT  DUP  1.  1.  SUB  ""  SREPL
          R−−>I  !  ROT  SWAP  /  SWAP
    END  DROP >>

XX1  #8566h  63.
<< DUP  SIZE  R−−>I  !
    WHILE  SWAP  DUP  NUM  CHR  ""  SREPL
          ROT  OVER  R−−>I  !  ROT
    REPEAT  /
    END  UNPICK >>

XX2  #4F57h  74.5
<< 1  SWAP  DUP  SIZE  1.
    FOR  j  DUP  NUM  CHR  ""  SREPL
          ROT  j  R−−>I  PICK3  R−−>I  COMB  *  UNROT  NEG
    STEP  DROP >>

XX3  #1D0Fh  80.5
<< 1  OVER  SIZE  R−−>I  ROT
    DO  DUP  NUM  CHR  ""  SREPL  ROT  OVER  R−−>I
          DUP2  COMB  6.  ROLL  *  5.  ROLLD  −  UNROT
    UNTIL  NOT
    END  DROP2 >>

XX4  #1564h  80.5
<< 1  0  ROT  1.  OVER  SIZE
    START  DUP  NUM  CHR  ""  SREPL  R−−>I  ROT  OVER  +
          UNROT  PICK3  OVER  COMB  5.  ROLL  *  4.  ROLLD
    STEP  DROP2 >>
```

There are two main ideas here and a variety of useful techniques. First, the command SREPL (new with the HP 49G) can be used to count how often each letter occurs in the string. SREPL not only replaces each occurrence of the substring in level 2 of the string in level 3 with the string in level 1 but also counts the number of times the replacement occurs. For example one has

"ABCDBC" "BC" "X" −−> "AXDX" 2.

David Hayden remarked afterwards that this counting is undocumented. The command itself is certainly described in the AUR but with an incorrect (incomplete) output listed. (Of course anyone who has used

the command would be aware of this.) The second idea is that these multinomial coefficients can otherwise be viewed (in many ways), and be more effectively calculated, as the product of binomial coefficients – for example:

(a+b+c)! / a! b! c! = COMB (a+b+c , a) * COMB (b+c , b) * COMB (c , c)

Of the six actual submissions only Jacob Wall used the first idea and only Roger Hill the second. (David Hayden calculated these numbers progressively without using either ! or COMB.) CB2, CB3 and XX1 all use the first idea – indeed XX1 is essentially the same as CB3 shortened by 15 bytes worth of technical sleights of hand. Using both ideas together leads to my programs XX2-XX4 (and many others) with their resulting better performances.

Note that JW4, CB3 and XX1-XX4 all work for arbitrary strings of any characters, not just those containing only A through Z. The length of the strings (after a certain point) does not affect comparative speeds all that much – the number of distinct characters they contain does, however. Testing these six programs on strings of 600 characters which cycle through all 256 characters (the output has 1290 digits) leads to the following comparison with the previous results where only the 26 letters A through Z were used.

(Strings of length 600)			26 different chars		256 different chars	
JW4	555h	141.	23.279	3282.4	62.395	8797.7
CB3	D71Ch	78.	18.120	1413.4	39.237	3060.5
XX1	8566h	63.	18.106	1140.7	39.101	2463.4
XX2	4F57h	74.5	8.540	636.3	24.974	1860.5
XX3	1D0Fh	80.5	8.233	662.7	20.767	1671.8
XX4	1564h	80.5	8.345	671.8	20.543	1653.7

It is noted that the relative performance (bytes*sec) of the three programs XX2-XX4 is exactly reversed when the full range of characters is used. Indeed the reason the problem was posed in the first place with the restriction to the 26 letters A-Z was not only to allow additional methods (used by five of the six entries) but also that by so doing a winning program (XX2) more clearly emerges. With regard to XX4 (the best in the last comparison), years ago there was a raging debate as to whether START…STEP (vs. FOR…STEP) could have any use whatever.

As the person responsible for this contest on this occasion I wish to thank all who took the trouble to enter (sacrificing precious sleep, time and energy during a very busy conference) and to reiterate that all six (plus two) entries were interesting – and instructive for anyone who cares to examine them.

O. T. Postscript: About this word "ANTITRINITARIAN" – it is simply the longest word I know containing no unrepeated letters. Needless to say I would be very interested to learn of any such word which is longer.

Programming contest

Every HHC has to have a programming contest. We conducted an RPL RPN Programming Contest for the HP 50g (conducted by Bill Butler) and then a contest for legacy RPN machines (Gene Wright). see appendix A for the Contest details.

The winner of the legacy RPN contest used the WP 34s. Code:

```
001 Rv                  013 RCL 01
002 Rv                  014 X^2
003 STO 01              015 -
004 DSE 01              016 SQRT
005 GTO 02              017 CEIL
006 GTO 03              018 STO+00
007 LBL 02              019 DSE 01
008 STO 00              020 GTO 01
009 X^2                 021 RCL 00
010 STO 02              022 LBL 03
011 LBL 01              023 4
012 RCL 02              024 x
```

The execution time for a radius of 5000 was about 28 seconds.

After the conference, solutions were posted on the HP Museum forum that were faster and for older machines. For reference the HP 67 found the answer for a radius of 5000 in about 1.4 hours.

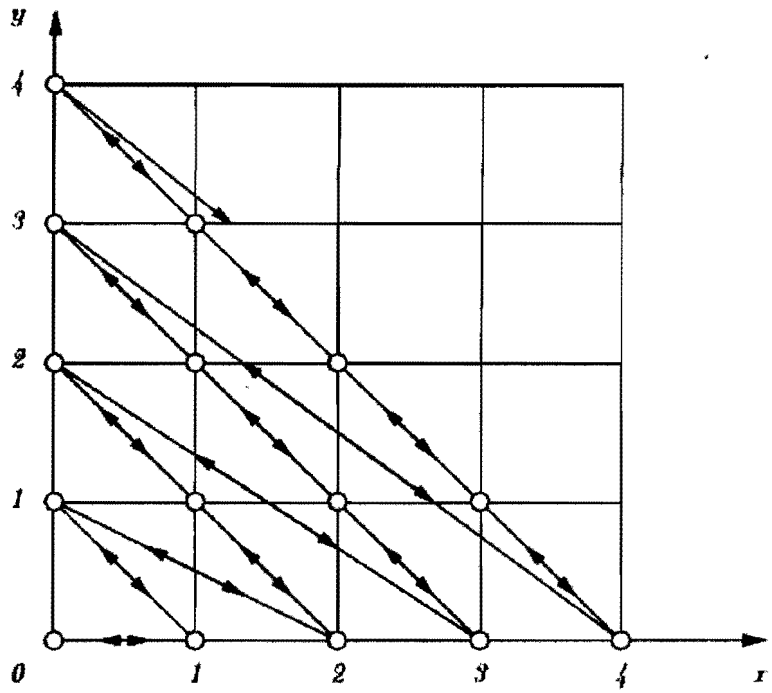
The fastest program posted to the museum was for the WP 34S. It solved the 5000 radius problem in just under 2 seconds, as it was found that integer mode on the WP 34S worked much faster.

```
001 BASE 10             014 RCL- Y
002 RCL Z               015 RCLx Y
003 FILL                016 SQRT
004 STO+ Z              017 FS? C
005 RCLx X              018 INC X
006 2                   019 SL 1
007 /                   020 STO+ Z
008 SQRT                021 DROP
009 INC X               022 DSE X
010 STO Z               023 BACK 10
011 STOx Z              024 4
012 -                   025 RCLx Z
013 RCL T               026 DECM
```

HHC 2012 RPN Programming Contest

September 22-23, 2012, Nashville

Problem Description: In the diagram below, each small circle has non-negative integer coordinates in the usual Cartesian coordinate system. You can move from one circle to another following the path denoted by the arrow symbols.



To move from (0,3) to (3,0), you have to pass through (1,2) and (2,1) then you arrive at (3,0), so this journey takes 3 "steps."

In this problem, you must compute the minimal number of steps needed to go from a given source circle to a given destination circle.

Input: Each test case consists of three integers, x_1, y_1, x_2, y_2 where each value is less than 1,000,000. These will be loaded as follows: x_1 ENTER y_1 ENTER x_2 ENTER y_2 then R/S.

Output: Return the minimum number of steps required with the sign of the answer indicated forward steps (positive) or backward steps (negative).

Sample Cases:
(A) Input of 1 ENTER 0 ENTER 1 ENTER 1 R/S should return 3.
(B) Input of 4 ENTER 0 ENTER 3 ENTER 0 R/S should return -4.

Machines Eligible: This contest is open to any and all RPN machines: 15c, 15c+, 15c LE, 34S, 41CL, 42S, 67, 65, etc. RPL users are welcome to try the problem, but this is for RPN machines only.

The winner will be the program that a) returns correct answers, b) has the shortest number of steps x speed in seconds, or c) if the speed is in the judge's sole opinion, nearly identical, the shortest routine.

Rules: (aka the fine print)

1) The decision of the judge is FINAL. No appeals are allowed to anyone or anything.
2) The purpose of this contest is to have fun and learn.
3) At least two contestants must submit an entry.
4) No custom built ROM or machine code can be built and used for this problem. Any already existing functionality in the machine is ok.
5) You must submit a machine with your program already keyed in to the judge AS WELL as a legible listing of your program with your name on the listing AND the machine. Machines with no names that are given to the judge are assumed to be gifts to the judge. Thank you!
6) Submission must be made by the end of the contest (Time is TBA).
7) Assume the program will start running with step 001 and/or a R/S.
8) By submitting a program, you agree to allow it to be shared with the community.
9) This is a contest between individuals, not teams. One submittal <> one person.
10) You may not access the internet for any help in any fashion. Do not cheat in any way. Do not check the HP Museum Forum either.
11) You must be present to win.
12) If a point is unclear, ask immediately. No excuses for ignorance. Clarifications will be shared with the entire group during the conference.
13) Assume default machine settings. Your program must stop with the default settings in place.
14) Winner will be the program with the fastest times over the test cases giving correct results. If in the judge's sole discretion, two entries are "about the same speed," the winner will be the shortest routine. In case of a tie, the most elegant solution (according to the judge) wins.
15) The purpose of this contest is to learn and have fun. Happy Programming.

HHC 2012 RPL Programming Contest

September 22-23, 2012, Nashville

Problem Description: The diameter of a set of points on the plain is the distance between its two most widely separated points. For example, the diameter of this set of points (1,1) (0,0) (2,3) (3,4) (1,0) is 5, which is the distance between (0,0) and (3,4). Given a set of points, compute its diameter.

Input: Each test case consists a list of up to 10 pairs of numbers. Each value in the list will be less than 10,000 in absolute value. The list will contain at least one pair of numbers and will always contain a multiple of 2 numbers, i.e., there will not be 3 values in the list.

Output: Return the diameter.

Sample Case: Input of { 1 1 0 0 2 3 3 4 1 0 } should return 5.

Machines Eligible: This contest is open to any and all RPL machines.

Rules: (aka the fine print)

1) The decision of the judge is FINAL. No appeals are allowed to anyone or anything.
2) The purpose of this contest is to have fun and learn.
3) At least two contestants must submit an entry.
4) No custom built ROM or machine code can be built and used for this problem. Any already existing functionality in the machine is ok. Sysevals, etc are allowed.
5) Your program must be transferred to the judge's machine under some identifying three-letter name before the announced deadline and you must also submit a legible listing of your program with your name on the listing.
6) Submission must be made by the end of the contest (Time is TBA).
7) By submitting a program, you agree to allow it to be shared with the community.
8) This is a contest between individuals, not teams. One submittal <> one person.
9) You may not access the internet for any help in any fashion. Do not cheat in any way. Do not check the HP Museum Forum either.
10) You must be present to win.
11) If a point is unclear, ask immediately. No excuses for ignorance. Clarifications will be shared with the entire group during the conference.
12) Assume default machine settings. Your program must stop with the default settings in place.
13) The winning program will be the one for which size*speed (bytes*sec) is least, where the speed of execution will be determined for one or more test cases chosen by the judge.
14) The program must be a (self-contained) single object in user code which does not call it self by name.
15) Default flag settings (except for flag –95) are assumed and must be restored if changed.
16) The stack, apart from input and output, must be left as found.
17) The program must not contain KILL or otherwise interfere with the programmatic testing and evaluation of submissions, i.e., you cannot delete everything on the judge's machine except your own program!
18) Happy Programming.

HHC 2013 RPN Programming Contest

September 21-22, 2013, Fort Collins, Colorado

Problem Description: All fractions written in octal (base 8) notation may be expressed exactly in decimal notation. For example, 0.75 in octal is 0.953125 in decimal. Specifically, a numeral requiring N octal digits to the right of the octal point may always be written as a decimal number with no more than 3N digits to the right of the point. The reverse is not always true.

Input: Each test case will consist of a value in octal between 0 and 1, inclusive.

Output: Display the base-10 decimal equivalent to the limits of your machine.

Sample Cases:
(A) 0.5 R/S should return 0.625.
(B) 0.75 R/S should return 0.953125.

Machines Eligible: This contest is open to any and all RPN machines: 15c, 15c+, 15c LE, 34S, 41CL, 42S, 67, 65, etc. RPL users are welcome to try the problem, but this is for RPN machines only.

The winner will be the program that a) returns correct answers, b) has the shortest number of steps x speed in seconds, or c) if the speed is in the judge's sole opinion, nearly identical, the shortest routine.

Rules: (aka the fine print)

1) The decision of the judge is FINAL. No appeals are allowed to anyone or anything.
2) The purpose of this contest is to have fun and learn and at least two contestants must submit an entry.
3) The speed results for all entries will be "normed" by counting the ticks for a loop. This loop program object will be loaded into YOUR machine by the judge.
4) No custom built ROM or machine code can be built and used for this problem. Any already existing functionality in the machine is ok.
5) You must submit a machine with your program already keyed in to the judge AS WELL as a legible listing of your program with your name on the listing AND the machine. Machines with no names that are given to the judge are assumed to be gifts to the judge. Thank you!
6) Submission must be made by the end of the contest (Time is TBA).
7) Assume the program will start running with step 001 and/or a R/S.
8) By submitting a program, you agree to allow it to be shared with the community.
9) This is a contest between individuals, not teams. One submittal <> one person.
10) You may not access the internet for any help in any fashion. Do not cheat in any way. Do not check the HP Museum Forum either.
11) You must be present to win.
12) If a point is unclear, ask immediately. No excuses for ignorance. Clarifications will be shared with the entire group during the conference.
13) Assume default machine settings. Your program must stop with the default settings in place.
14) Winner will be the program with the fastest times over the test cases giving correct results. If in the judge's sole discretion, two entries are "about the same speed," the winner will be the shortest routine. In case of a tie, the most elegant solution (according to the judge) wins.
15) The purpose of this contest is to learn and have fun. Happy Programming.

HHC 2013 RPL Programming Contest

September 21-22, 2013, Fort Collins, Colorado

Problem Description: Normal base-10 addition involves a carrying step whenever two digits sum to 10 or greater. For example, in 23 + 49 = 72, the 3 + 9 involves carrying a 1 to the tens unit. In false addition, any numbers that would be carried are simply dropped. So 23 + 49 = 62, since 3 + 9 = 12 (giving the 2 in the units place), and 2 + 4 = 6 (ignoring the carried 1).

Input: Integer a in level 3, Integer b in level 2, the base in level 1 of the stack. Note: these will be keyed as real numbers but will not have any values after the decimal point. Bases entered will be 2, 8, 10 or 16.

Output: The result of false addition of the real numbers a and b in the input base. Your program should work regardless of the base of the machine when the program is run and your program should stop in the entered base. This is the only allowable change to the machine's status.

Sample Cases:
1) 499 ENTER 861 ENTER 10. After executing your program, the output should be 250.
2) 654 ENTER 456 ENTER 8. After executing your program, the output should be 222.

Machines Eligible: This contest is open to any and all RPL-style machines.

Rules: (aka the fine print)

1) The decision of the judge is FINAL. No appeals are allowed to anyone or anything.
2) The purpose of this contest is to have fun and learn and at least two contestants must submit an entry.
3) The speed results for all entries will be "normed" by counting the ticks for a loop. This loop program object will be loaded into YOUR machine by the judge.
4) No custom built ROM or machine code can be built and used for this problem. Any already existing functionality in the machine is ok. Sysevals, etc are allowed. Your program must be ONE OBJECT. Everything must be self-contained in this one object. No pre-storing of constants, etc. is allowed.
5) You must also submit a legible listing of your program with your name on the listing. Your program must run on your own machine.
6) Submission must be made by the end of the contest (Time is TBA).
7) By submitting a program, you agree to allow it to be shared with the community.
8) This is a contest between individuals, not teams. One submittal <> one person.
9) You may not access the internet for any help in any fashion. Do not cheat in any way. Do not check the HP Museum Forum either.
10) You must be present to win.
11) If a point is unclear, ask immediately. No excuses for ignorance. Clarifications will be shared with the entire group during the conference.
12) Assume default machine settings. Your program must stop with the default settings in place except as noted above in the problem description. Default flag settings (except for flag –95) are assumed and must be restored if changed again except as noted above.
13) The winning program will be the one for which size*speed (bytes*sec) is least, where the speed of execution will be determined for one or more test cases chosen by the judge.
14) The program must be a (self-contained) single object in user code which does not call it self by name.
15) The stack, apart from input and output, must be left as found.
17) The program must not contain KILL or otherwise interfere with the programmatic testing and evaluation of submissions, i.e., you cannot delete everything on the judge's machine except your own program!
18) Happy Programming.

HHC 2014 Programming Contest
PRIME DATE PAIRS (PDP's)
One contest, 3 winners (RPN, RPL, & PPL)

A "Prime Date" is a date which, when written in yyyymmdd form, is a prime number, e.g. 27 Sept 2014, because 20140927 is a prime number. It is the next prime date after HHC 2014.

A "Prime Date Pair" (hereafter PDP) are two prime dates which are consecutive calendar dates, e.g. 19500331 (31 March 1950) and 19500401 (1 April 1950). Note that, unlike so-called "prime pairs" which differ by 2 (e.g. 11 and 13), PDP's differ by 1 calendar day. A "PDP date" is any date which belongs to a PDP.

Programming Contest: Write a program which, given any year between 1583 and 9999, outputs all the PDP dates in that year (and only in that year) in yyyymmdd format, or mm.ddyyyy format, or dd.mmyyyy format, whichever you prefer.

Notes:

1. The winners are the 3 shortest programs: 1 in RPN, 1 in RPL, and 1 in PPL. RPN program size will be counted in steps. RPL and PPL program size will be counted in bytes. This process is somewhat imprecise, so the judge's decision will be final.

2. Some models do not contain native primality testing functions. Therefore, programs for these models may call an external primality testing subroutine (e.g. NP in the PPC ROM, or any other program or function in the machine), to help minimize the size of your program. The purpose of this rule is to make it a fair contest, because the size of the external primality tester will not be included in the calculation of the size of your program. However, the primality testing subroutine must ONLY test primality. If you hide chunks of your main program in the primality testing program, you will be disqualified. Obviously.

3. Speed is irrelevant; elegantly efficient code is the goal of this contest. However, all programs must return the correct answers to qualify, so programs that run too long to be judged will be disqualified.

4. As always, any program which violates the goal of elegant code packing (according to the sole discretion of the contest judge) will be disqualified. For example, embedding commands in a string, then executing the string, just to save a few bytes, is the antithesis of elegance. Bonus points for making the judge gasp in awe. Negative points for making the judge gasp in horror.

5. The judge reserves the right to input any years from 1583 through 9999. An already-existing complete list of PDP's from 1583 to 9999 will be used for judging.

HOMEWORK PROBLEMS

1. Easy: Prove that there cannot be a PDT (Prime Date Triple).

2. Harder: If everybody lived forever after being born, what percent of the population would never have prime birthdays?

3. Difficult: Some years contain an odd number of PDP dates, e.g. 1978 (Dec 31st only) and 1979 (Jan 1st , Jan 31st, and Feb 1st only). Prove that there cannot be 4 consecutive calendar years which all contain exactly an odd number of PDP dates.

HHC 2015 RPN Programming Contest

This time, it's personal

September 26-27, 2015, Nashville Tennessee

Problem Description: For over 30 years, I have written and tweaked an HP-41C aka FOCAL program that plays and scores the game of Yahtzee. This game is included in the HP 41CL Funstuff rom with the label YATZ. However, that version, first published in a 1985 PPC Journal issue, has bugs in one of the scoring labels. Time to put that to bed forever with a bugless short routine from the winner of this contest!

The game of Yahtzee requires the examination of several sets of results for five dice to allow for proper scoring, such as three of a kind, full house, etc. The hardest to detect in a calculator program in my experience is the small straight, defined as four sequential values in the group of five dice.

The complete list of all 16 possible small straights is shown below. These have been sorted in ascending order.

11234	12344	13456	23445	33456	34566
12234	12345	22345	23455	34456	
12334	12346	23345	23456	34556	

Input: The values of the five dice (integers 1, 2, 3, 4, 5, or 6) will be stored in memories 1 through 5 in a random order.

Output: If a small straight is present, return a 0. If a small straight is not present, return anything other than a zero. Preserving the stack otherwise is irrelevant.

Sample Cases:
(A) 12345 stored in 1 through 5 and then R/S should return 0.
(B) 11123 stored in 1 through 5 and then R/S should return something other than 0, even if stored as 21131.
(C) 12356, 11111, 41623, 22344, etc., and then R/S should return something (anything) other than 0.

Machines Eligible: This contest is open to the following RPN machines: HP-41C, 41CL, 42S, 34S – or other RPN machines subject to the following: whatever machine you use, it must only include standard HP 41 functions. No function present in the 34S or 42S or 41CL only is allowed. Standard vanilla HP 41 programming functions please – no 41CX, X-functions or synthetic programming allowed either. If you can make the routine shorter using non-vanilla HP 41 functions, please let the judge know, but the contest must be vanilla. You may not win the contest, but you'll have the judge's gratitude.

The winning routine returns correct answers and has the lowest number of bytes as determined by the routine being keyed into an HP 41C. If you use a non-HP 41c machine, it is the byte count on the 41C that matters. Speed is not of the essence.

Rules: (aka the fine print)

1) The decision of the judge is FINAL. No appeals are allowed to anyone or anything.
2) The purpose of this contest is to have fun and learn and at least two contestants must submit an entry.
3) No custom-built ROM or machine code can be built and used for this problem. Any already existing functionality in the machine is ok.
5) You must submit a machine with your program already keyed in to the judge AS WELL as a legible listing of your program with your name on the listing AND the machine. Provide byte count if possible. Machines with no names that are given to the judge are assumed to be gifts to the judge.
6) Submission must be made by the end of the contest (Time is TBA).
7) Assume the program will start running with step 001 and successive runs must work with pressing R/S.
8) By submitting a program, you agree to allow it to be shared with the community.
9) This is a contest between individuals, not teams. One submittal <> one person.
10) You may not access the internet for any help in any fashion. Do not cheat in any way. Do not check the HP Museum Forum either.
11) You must be present to win.
12) If a point is unclear, ask immediately. No excuses for ignorance. Clarifications will be shared with the entire group during the conference.
13) Assume default machine settings. Your program must stop with the default settings in place, unless they are changed by the provided sorting routines.
14) The routine must start with a LBL RR. If you need to use a sorting routine, two options are provided below. The sorting routine (if used) should be external to your program listing. Any calls to a sorting routine are included in your byte count, but the sorting routine itself is not. The sorting routine must have a four-character label to keep things consistent.
15) The purpose of this contest is to learn and have fun. Happy Programming.
16) The shortest routine the judge has as of 5/18/2015 is 72 bytes on an HP 41C and that includes 12 bytes for two XEQ SORT subroutine calls and the global RR label. Your routine may need no sort calls, one sort call, two or more. That's up to you.

Sample sorting routines.

Routine one: To sort registers 1 – 5, key XEQ SORT. Any calls to this routine count in your routine size total. Thanks to Jean-Marc Baillard for this sorting routine from the hpmusem.org software library. Routine ONE requires the ability to handle functions such as ISG L and X<> IND L.

Routine two: To sort registers 1 – 5, key XEQ SORT. This one is "fun" in that if an exchange between memories is made to sort the registers, you will see the 0 flag indicator turn on and off in the display on an HP 41. This is suitable for any RPN machines that cannot do the indirect and ISG commands used in Routine ONE. For more "fun", change CF 00 to DEG and SF 00 to RAD throughout the routine. Then replace FS? 00 with FS? 43. See what happens as it sorts. Routine TWO should work on most RPN calculators

Improvements to the sorting routine (if any) are a distraction. These are not part of the contest. The actual YZ Yahtzee program will call an MCode SORT command. This is provided solely to give a level playing field and to avoid asking attendees to reinvent the wheel.

Routine ONE

```
01 LBL "SORT"
02 1.005
03 SIGN
04 LBL 01
05 LASTX
06 LASTX
07 RCL IND L
08 LBL 02
09 RCL IND Y
10 X>Y?
11 GTO 03
12 X<>Y
13 LASTX
14 +
15 LBL 03
16 RDN
17 ISG Y
18 GTO 02
19 X<> IND L
20 STO IND Z
21 ISG L
22 GTO 01
23 END
```

Routine TWO

```
01  LBL "SORT"
02  LBL 01
03  CF 00
04  RCL 04
05  RCL 03
06  RCL 02
07  RCL 01
08  X>Y?
09  SF 00
10  X>Y?
11  X<>Y
12  STO 01
13  RDN
14  X>Y?
15  SF 00
16  X>Y?
17  X<>Y
18  STO 02
19  RDN
20  X>Y?
21  SF 00
22  X>Y?
23  X<>Y
24  STO 03
25  RDN
26  RCL 05
27  X<Y?
28  SF 00
29  X<Y?
30  X<>Y
31  STO 05
32  X<>Y
33  STO 04
34  FS? 00
35  GTO 01
36  END
```

HHC 2015 RPL Programming Contest

Write a program for the HP50g in RPN Mode which tests whether a non-empty list of positive integers (type 28 objects) consists of distinct primes, replacing such a list in level 1 of the stack with 1 (or 1.) if true and 0 (or 0.) if false.

The winning program will be the fastest, with the speed of execution averaged over a few input lists selected by the judges, each containing at least 100 positive integers.

The usual rules apply:

- The program must be a (self-contained) single object in user code which does not call itself by name
- Default flag settings (except for flag –95) are assumed and must be restored if changed
- The stack, apart from input and output, must be left as found
- The program must not contain KILL or otherwise interfere with the programmatic testing and evaluation of submissions
- Your program must be transferred to the judges' machine under some identifying 3-letter name before the announced deadline
- The purpose of the contest is to have fun and the decision of the judges is final

HHC 2015

HHC 2016 RPN Programming Contest

September 17-18, 2016, Fort Collins, Colorado

Problem Description: Let f(k,n) denote the number of positive k-digit integers whose digits add up to n. Example, f(4,5) = 35, since 35 is the number of 4-digit positive integers whose digits add up to 5, as shown below.

```
1004 1013 1022 1031 1040 1103 1112
1121 1130 1202 1211 1220 1301 1310
1400 2003 2012 2021 2030 2102 2120
2111 2201 2210 2300 3002 3011 3020
3101 3110 3200 4001 4010 4100 5000
```

Write a program that prompts computes f(k,n). You may assume that 1 <= k <= 9 and that 1 <= n <= 100. (in reality, n cannot be larger than 81, but let's say 100 here).

Input: k ENTER n.
Output: Display the value of f(k,n).

Sample Cases. Check your program on these before submitting. Beware long run times. Answers can be > 10 million.

(A) 6 ENTER 34 R/S. Answer displayed is 33787.
(B) 3 ENTER 27 R/S. Answer displayed is 1.
(C) 5 ENTER 0 R/S. Answer displayed is 0. Note: n must be 1 <= n <= 100.
(D) 12 ENTER 5 R/S. Answer displayed is 0. Note: k must be 1 <= k <= 9.
(E) 5 ENTER 20 R/S. Answer displayed is 4998.

Machines Eligible: Contest is for the WP-34S only*. If you don't have a 34S here, download the free app and start coding.

The winning routine returns correct answers and has the lowest number of bytes or steps. Shortest routine wins. Memories used will be assessed an 8 byte cost added to length of routine. Same program length, but use an extra memory? +8.

*__Note:__ However, a special prize will be given to anyone who creates a working solution on the SR-56 calculator. The Judge has one available for attempts to do this.

Rules: (aka the fine print)

1) The decision of the judge is FINAL. No appeals are allowed to anyone or anything.
2) The purpose of this contest is to have fun and learn and at least two contestants must submit an entry.
3) No custom-built ROM or machine code can be built and used for this problem. Any already existing functionality in the machine is ok. No calls to library programs, etc. All code should be in the program itself.
4) You must submit a machine with your program already keyed in to the judge AS WELL as a legible listing of your program with your name on the listing AND the machine. Provide byte / step count if possible. Machines with no names that are given to the judge are assumed to be gifts to the judge. Thank you very much.
5) Submission must be made by the end of the contest (Time is TBA). You must be present to win.
6) The program should start running with step 001 (no label) and successive runs must work by pressing R/S.
7) By submitting a program, you agree to allow it to be shared with the community.
8) This is a contest between individuals, not teams. One submittal <> one person.
9) You may not access the internet for any help in any fashion. Do not cheat in any way. Do not check the HP Museum Forum either.
10) If a point is unclear, ask immediately. No excuses for ignorance. Clarifications will be shared with the entire group during the conference.
11) Assume default machine settings. Your program must stop with the default settings in place.
12) Successive runs must allow for new inputs and R/S to continue with the next run. Execution should start at step 01. The original stack and memory contents do not need to be saved.
13) You only need to display the value of f(k,n). You do not need to count or show or save the actual values of f(k,n) that satisfy the conditions. The numbers shown at the top are for illustrative purposes only.
14) The purpose of this contest is to learn and have fun. Happy Programming.

HHC 2016 RPL Programming Contest

September 17-18, 2016, Fort Collins, Colorado

Problem Description: The Maximum Character Separation of a string is the greatest difference between any two identical letters in the string. For example, the maximum character separation of HIPPOPOTAMUS is 3, because that is the distance from the first P to the last P (and no other two identical letters are separated by a greater distance).

H I P P O P O T A M U S

1 2 3 4 5 6 7 8 9 10 11 12

Write a program that accepts a character string on an otherwise clear stack and outputs the maximum character separation. You may assume capital letters only and a non-null string.

Input: string.
Output: value of maximum character separation.

Sample Cases: Store your program as object 'MCS'

(A) "HIPPOPOTAMUS" ENTER and press MCS menu label. Answer displayed is 3.
(B) "RATTLESNAKE" ENTER and press MCS menu label. Answer displayed is 7.
(C) "PORCUPINE" ENTER and press MCS menu label. Answer displayed is 5.
(D) "FLAMINGO" ENTER and press MCS menu label. Answer displayed is 0. Note: No repeated letters.

Machines Eligible: Contest is for RPL machines only. The 48G through 50g are allowable. No 28C. Sorry!

The winning routine returns correct answers and has the lowest number of bytes. Shortest routine wins.

Rules: (aka the fine print)

1) The decision of the judge is FINAL. No appeals are allowed to anyone or anything.
2) The purpose of this contest is to have fun and learn and at least two contestants must submit an entry.
3) Any already existing functionality in the machine is ok. The program must be keyable on a clear machine.
4) You must submit a machine with your program already keyed in to the judge AS WELL as a legible listing of your program with your name on the listing AND the machine. Provide byte count. Machines with no names that are given to the judge are assumed to be gifts to the judge.
5) Submission must be made by the end of the contest (Time is TBA).
6) By submitting a program, you agree to allow it to be shared with the community.
7) This is a contest between individuals, not teams. One submittal <> one person.
8) You may not access the internet for any help in any fashion. Do not cheat in any way. Do not check the HP Museum Forum either.
9) You must be present to win.
10) If a point is unclear, ask immediately. No excuses for ignorance. Clarifications will be shared with the entire group during the conference.
11) Assume default machine settings. Your program must stop with the default settings in place. Assume a clear stack other than the input string.
12) The purpose of this contest is to learn and have fun. Happy Programming.