HP Forum Archive 14

[Return to Index | Top of Index]

Short & Sweet Mathematical Challenge #7

*Message #1 Posted by **Valentin Albillo** on 5 July 2004, 9:27 a.m.*

Hi all, a new month begets a new S&S Math Challenge, #7 to be precise.

This time you'll have to use to the max all your math intuition, skills, a lot of sleuthing, and a respectable amount of programming ingenuity though, as always, the required program won't be neither too difficult nor too long.

I can assure you that the answer to this particular challenge will take you *absolutely by surprise*, you'll be *enthralled* by the discovery process, then absolutely *delighted* with the unexpected outcome. Matter of fact, you might want to tackle this challenge all by yourself, refraining from reading any posts in this thread till you find the solution on your own, so as to avoid spoiling the (considerable) pleasure. This is truly one of a kind, trust me !

So, it seems this would be reward enough for any HP/Math fan, right ?

Well, no. Just in case this isn't enough for you, there's yet another interesting (if material) reward: a high-quality 1280 x 1024 image of this fabulous classic "poster", scanned by myself, depicting three Classical and Woodstock HP machines against an ancient pyramidal background, have a look at this small thumbnail:

This magnificent 1280 x 1024 image will be awarded to the first person (or persons, eventually) who comes out with a working program and correct answers to the questions raised by

The Challenge

This Challenge is structured in two parts. *First*, you must write a program to compute a given function. *Then*, you must use your program *and* intuition to answer a number of preparatory, training questions about the function, ending in a **Big Question[tm]** which will require all your sleuth abilities as well as data gathered from the answers to the training questions.

First part: The Program

Use your favorite HP calculator (most any will do) to implement the following function **h(n)**, defined for integer n greater than 0 as follows:

```
h(1)    = 1
h(3)    = 3
h(2n)   = h(n)
h(4n+1) = 2*h(2n+1)-h(n)
h(4n+3) = 3*h(2n+1)-2*h(n)
```

Your program must accept **n** as input and return **h(n)** as output, where n is an arbitrary integer greater than 0.

If you succeed, you'll be in a most favorable position to answer the following *training* questions, which will increase your chances of finding a correct answer to the ultimate last question.

Second part: The Questions

Use your program to compute **h(n)** and a little sleuthing to answer these training questions, *in order*. They are designed to led you effortlessly, step by step, to the **Big Question[tm]**. (m and n are always arbitrary integers greater than 0).

1. What's the value of $h(2^n)$? ditto $h(2^n + 1)$? $h(2^n - 1)$? $h(m * 2^n)$?
2. For which n is h(n) = 100 ?
3. For which n is h(n) even ?
4. If n is odd and m = h(n), what's the value of h(m) ?
5. What are the values of h(h(n)) ?
6. For which values of n do we get h(n) = n ?

Second part encore: The Big Question [tm]

Just what on Earth does h(n) to n ??

This is, describe the simplest relationship the result value has with the given argument. In English (or Spanish) you can state this relationship unambiguously using just ***three words***. Which ones ?

Notes:

In order to check that your program delivers correct results for h(n), you might try to check out the following random cases:

```
   n  h(n)
------------
  841  587
  642  261
   44   13
  821  691
  381  381
   17   17
  378  189
  226   71
   86   53
  742  413
  765  765
  514  257
```

Give it a try, I'm sure you'll enjoy it ! :-)

Best regards from V.

Re: Short & Sweet Mathematical Challenge #7

*Message #2 Posted by **J-F Garnier** on 5 July 2004, 12:34 p.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin,

First of all, my test program on my HP-71B. I used user-defined function and recursivity, close to the function definition:

```
10 DEF FNH(N)
20 IF N=1 THEN FNH=1 @ END
30 IF N=3 THEN FNH=3 @ END
40 IF MOD(N,2)=0 THEN FNH=FNH(N/2) @ END
50 IF MOD(N,4)=1 THEN FNH=2*FNH((N-1)/2+1)-FNH((N-1)/4) @ END
60 FNH=3*FNH((N-3)/2+1)-2*FNH((N-3)/4)
70 END DEF
```

Maybe the implementation on a RPN, limited-stack machine would be more tricky. It could be a challenge by itself.

I succeeded to find the relation between n and h(n), but I don't want to spoil the game right now. I will just say that fnh(168788912877) is 196856925369 (quite hard to check with the recursive method) and that the HP-71B Math ROM includes a convenient function to check this.

It was really a pleasure to spend some time on your challenge.

J-F

Re: Short & Sweet Mathematical Challenge #7

*Message #3 Posted by **Bram** on 5 July 2004, 1:32 p.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin,

I already found the relation between n and h(n) and my phrase would be E.A.R.
I'll explain this later, because I first have to write my program. I manually compiled a table for n=1..20 and I discovered the relation fairly quickly.
Knowing the relation I could easily write a non-recursive routine, but I will try to write a program as if I don't know anything about it yet.

Re: Short & Sweet Mathematical Challenge #7

*Message #4 Posted by **Bram** on 5 July 2004, 2:57 p.m.,*
in response to message #3 by Bram

Well, no program yet. Quite a challenge for an HP-32SII without recursion, but I won't wait for it to explain the E.A.R. in my previous post.

So here's the relation I've found

DON'T READ ON IF YOU DON'T YET WANT TO KNOW

DON'T READ ON IF YOU DON'T YET WANT TO KNOW

FINAL WARNING

Extract And Reverse is how I've put it.
Take n
Convert it to binary
Extract the part from the first 1 until the last 1 (so strip leading and trailing zeros)
Reverse this string of binary digit(s)
Convert back to decimal
Et voilá

And that explains why there aren't any even results; you'll always have a 1 as the LSB, hence an odd number.
And also why 2^n+1 doesn't change; two ones separated by zeros
And also why 2^n-1 doesn't change; all ones

Thanks Valentin, for this well spent night (local time is 21:00)

Re: Short & Sweet Mathematical Challenge #7

*Message #5 Posted by **Veli-Pekka Nousiainen** on 6 July 2004, 12:02 a.m.,*
in response to message #4 by Bram

```
I present only a HP-49G program (not perfect :)
<< -> n
   << n
      IF DUP 0 >
      THEN PUSH BIN R->B ->STR 3
OVER SIZE SUB SREV TAIL "#" SWAP
+ STR-> B->R R->I POP
      END n
   >> DROP
>>
BYTES => # 14775o 111.5
<< VPN >>
```

Re: Short & Sweet Mathematical Challenge #7

*Message #6 Posted by **Carl Nelson** on 9 July 2004, 2:48 a.m.,*
in response to message #5 by Veli-Pekka Nousiainen

For HP-48G:

<< BIN 64 STWS R->B #0b SWAP

WHILE DUP B->R REPEAT SR LASTARG #1b AND ROT SL OR SWAP END

DROP B->R >>

I also did a solution for the HP-11C, HP-12C, and HP-42S, but I don't have the listings handy here.

--Carl

Massimo's Last Theorem

*Message #7 Posted by **Massimo (Italy)** on 9 July 2004, 4:10 a.m.,*
in response to message #6 by Carl Nelson

> Quote:
>
> also did a solution for the HP-11C, HP-12C, and HP-42S, but I don't have the listings handy here

I found a clever solution for the HP-01 but the margin on my copy of this book by Diophantus is too small...

:-)

Re: Massimo's Last Theorem

*Message #8 Posted by **Ángel Martin** on 9 July 2004, 7:57 a.m.,*
in response to message #7 by Massimo (Italy)

You need to stop buying those chipo books with such small margins. Try the coffe-table edition next time !

Best, ÁM

Re: Massimo's Last Theorem

*Message #9 Posted by **Carl Nelson** on 10 July 2004, 12:52 a.m.,*
in response to message #8 by Ángel Martin

Wise guys, eh? :)

Here is the code for the HP-42S. It's the most compact. The 11C AND 12C are similar, except for the issue of labels (single letter/number on 11C, nonexistant on 12C, use line numbers for GTOs), and lack of MOD function on both (use 2 / frac 2 *)

LBL “HOFN”

0

STO 00

(ROLLDOWN)

LBL 01

X=0?

GTO 02

ENTER

ENTER

2

STO * 00

MOD

STO + 00

(ROLLDOWN)

2

/

IP

GTO 01

LBL 02

RCL 00

R/S

Re: Massimo's Last Theorem

*Message #10 Posted by **Massimo (Italy)** on 10 July 2004, 7:43 a.m.,*
in response to message #9 by Carl Nelson

> Quote:
>
> Wise guys, eh? :)

No offence taken, I hope. I just couldn't resist after reading your original post.

Massimo

Re: Short & Sweet Mathematical Challenge #7

*Message #11 Posted by **RMillán** on 5 July 2004, 7:32 p.m.,*
in response to message #1 by Valentin Albillo

Hi, Valentín:

While thinking about how to program this *h* function in a 15C, I'll post a simple program for the 16C which will compute the function:

```
001 LBL A
002 0
003 x<>y
004 LBL 0
005 x!=0
006 GTO 1
007 RDN
008 RTN
009 LBL 1
010 SR
011 x<>y
```

```
012 RLC
013 x<>y
014 GTO 0
```

It works in any wordsize, but of course 64 will give the maximum numeric range.

I suppose that using the 16C for this challenge is kind of cheating. Indeed, I did not begin, as was required, by programming the function in *my favorite HP calculator (most any will do)*; among the non-RPL calculators, only the 16C allows a trivial implementation.

The challenge is now for me to find a reasonably implementation in the 15C. Also, I'm not so sure I could express the relation (in English or in Spanish) in just three words.

Regards,
Rafael Millán.

Re: Short & Sweet Mathematical Challenge #7

*Message #12 Posted by **Cameron** on 6 July 2004, 5:04 a.m.,*
in response to message #1 by Valentin Albillo

I suspect the horse has bolted (judging from the number of responses). However I'll make my contribution and await for Valentin to declare a winning entry before I look at the others.

Normally I treat these challenges as a spectator sport. They're typically most illuminating but require a prowess in mathematics that I don't posess. The word that caught my eye on this one was *integer*. I know a thing or two about (the computer representation of) integers. So I read the challenge a little more closely than usual and decided I could produce a solution.

Valentin said to use my favorite HP, so I fired up my 16C simulator (it works on the real one too). Here's the program:

```
001     LBL FN_H        ;       n
002     [0]             ;       h=0     n
003     X-Y             ;       n       h
004     LBL LOOP        ;       n       h


; shift LSB out of n into carry { carry = n & 1; n >>= 1; }


005     SR              ;       n=n/2   h
006     X-Y             ;       h       n
```

```
; shift LSB into h from carry { h = h<<1 | carry; }


007     RLC             ;       h=RLC   n
008     X-Y             ;       n       h


; Any bits left in n?


009     X!=0?
010     GTO LOOP


; cleanup--result into X


011     X-Y             ;       h       0
012     (RTN)           ; not necessary if last in memory
```

And now the answers to the questions:

> Quote:
>
> 1. What's the value of h(2^n) ? ditto h($2^n + 1$) ? h($2^n - 1$) ? h($m * 2^n$) ?

```
1a: 1
1b: 2^n + 1
1c: 2^n - 1
1d: if m is odd, m else if m == 2^x, 1 else h(n)
```

> Quote:
>
> 2. For which n is h(n) = 100 ?

h(n) can never be 100.

> Quote:
>
> 3. For which n is h(n) even ?

h(n) can never be even.

> Quote:
>
> 4. If n is odd and m = h(n), what's the value of h(m) ?

n

> Quote:
>
> 5. What are the values of h(h(n)) ?

n

> Quote:
>
> 6. For which values of n do we get h(n) = n ?

$2^x - 1$

Before I answer the big question^{TM} I have to confess to having cheated. Valentin invited us:

> Quote:
>
> to check out the following random cases:

I looked carfully at those numbers. If indeed they were "random" and not carfully contrived, I could see the (bit) pattern that led to my conclusion about the defining property of h(n).

Now, the ultimate answer: right-justified bit reversal.

Of course that's not what Valentin was looking for. I've wracked my brains to weave the "Earth" hint into the answer but it eludes me. I am sure that someone who more clever with word-play will get it.

BTW, right-justified bit reversal is a technique that is used in permuted index generation. One applicatition that springs to mind is DFT. I'm sure Knuth wrote about it in *Sorting and Searching*.

Cameron

PS: a corollary challenge. Write a program to compute the probability that Valentin's (random) test cases were all right-justified reversals.

Re: Short & Sweet Mathematical Challenge #7

*Message #13 Posted by **J-F Garnier** on 6 July 2004, 8:08 a.m.,*
in response to message #12 by Cameron

My answer to question #6 would be 'n is a binary palindrome'.

Regarding DFT, I think that the reversal is NOT right-justified (otherwise no even index would be generated).

J-F

Re: Short & Sweet Mathematical Challenge #7

*Message #14 Posted by **Cameron** on 6 July 2004, 8:48 a.m.,*
in response to message #13 by J-F Garnier

> Quote:
>
> My answer to question #6 would be 'n is a binary palindrome'.

h(n) is a palindrome iff n is a palindrome.

> Quote:
>
> Regarding DFT, I think that the reversal is NOT right-justified (otherwise no even index would be generated).

D'oh! You are, of course, correct. Apologies to all for the misinformation.

Cameron

definition of binary palindrome?

*Message #15 Posted by **Cameron** on 7 July 2004, 3:43 p.m.,*
in response to message #13 by J-F Garnier

It occurs to me that my notion of palindrome may be different to everyone else's. As a computer guy, I tend to think of all bits (in a given parcel) as being significant. Consider this octet:

00001001

To my mind, that is *not* a palindrome although its right-justified reversal will have the same value. This demonstrates the flaw in my thinking. Since the leading zeros are numerically insignificant, it is a palindrome (although I would still contend that the octet *holds* a palindrome).

In light of this, your answer to question 6 would appear to be correct.

Comments?

Cameron

Ahhrrrg! A typo.

*Message #16 Posted by **Cameron** on 6 July 2004, 8:58 a.m.,*
in response to message #12 by Cameron

> Quote:
>
> ```
> Quote:
> ---------------------------------------------------------------------------
>  6. For which values of n do we get h(n) = n ?
> ---------------------------------------------------------------------------
> ```
>
> 2^x - 1

This should read:

2^{x+}/- 1

Serves me right for dinking with fancy formatting.

Cameron

Question#6

*Message #17 Posted by **Tizedes Csaba [Hungary]** on 6 July 2004, 10:51 a.m.,*

in response to message #16 by Cameron

I dont want to provoke - because I havent got answer - but what about n=381 and n=765...?!

Csaba

Ps.: I am working on my solution. I dont believe in this 'binary-palindrom'... Its just pure mathematics... ;) But not evident...

Re: Question#6

*Message #18 Posted by **Cameron** on 6 July 2004, 11:11 a.m.,*
in response to message #17 by Tizedes Csaba [Hungary]

Well spotted! <Gently lays queen on her side>

It's obvious now when I reconsider it. All the odd palindromes < 2^{max} - 1 must satisfy h(n) == n. Is there a simple expression that defines the series?

I'm off to bed. I'll check back in the morning.

Cameron

Re: Short & Sweet Mathematical Challenge #7

*Message #19 Posted by **Cameron** on 7 July 2004, 4:19 p.m.,*
in response to message #12 by Cameron

Obviously my answer to 5 is only true if n is odd. The series that results from h(h(n)) for even n is fascinating. However I don't know what to make of it.

Cameron

Re: Short & Sweet Mathematical Challenge #7

*Message #20 Posted by **W Rice** on 7 July 2004, 8:37 p.m.,*
in response to message #19 by Cameron

For n even, it's the first odd number you encounter after continually dividing by 2.

No solution yet, just two equation...

*Message #21 Posted by **Tizedes Csaba [Hungary]** on 8 July 2004, 7:36 a.m.,*
in response to message #1 by Valentin Albillo

It's trivial, but maybe(???) this will help somebody:

```
[[ 3  -2 ]  * [[ h(4*n+1) ]  =  [[   h(n)   ]
 [ 2  -1 ]]    [ h(4*n+3) ]]     [ h(2*n+1) ]]
```

Csaba

S&SMC #7: My Solutions And Comments [LOOOOONG]

*Message #22 Posted by **Valentin Albillo** on 8 July 2004, 8:49 a.m.,*
in response to message #1 by Valentin Albillo

Thanks to all of you interested in my Challenge, whether 'posters' or 'lurkers'. Questions have been answered correctly and some working programs have been produced, though some of you 'cheated' and found the answers by analyzing the test cases I gave, instead of actually implementing h(n) as defined, which was the intended exercise. Well, so much for giving test cases, now for my solution and comments:

Note: All the code herein is for the HP-71B because its English-like BASIC syntax makes it very easy to understand and vey easy to translate to RPN/RPL, while the reverse isn't exactly true. Also, even if you haven't got an HP-71B, you can try and run all the code and examples featured here by using Emu71, Jean-François Garnier's superb HP-71B emulator which you can download for free from that link.

First part: The Program

The Challenge required you to implement h(n) defined for integer n > 0 as follows:

```
h(1)    = 1
h(3)    = 3
h(2n)   = h(n)
h(4n+1) = 2*h(2n+1)-h(n)
h(4n+3) = 3*h(2n+1)-2*h(n)
```

If your HP machine does natively support *recursion*, this is very easy to implement, like this HP-71B version, which defines a user-defined function FNH(N) which computes h(n) by making recursive calls to itself:

```
10 DEF FNH(N) @ IF N=1 OR N=3 THEN FNH=N @ END
20 ON RMD(N,4)+1 GOTO 30,40,30,50
30 FNH=FNH(N/2) @ END
```

```
40 FNH=2*FNH((N+1)/2)-FNH((N-1)/4) @ END
50 FNH=3*FNH((N-1)/2)-2*FNH((N-3)/4) @ END DEF
```

Implementing h(n) in an HP calc without recursion is much more *tricky*, and usually requires dealing with stacks to keep the parameters and returns for the ever deeper recursive calls, but once you get the knack of it, it's actually fairly straightforward.

However, for this particular challenge, where implementing the function is just for the sake of using it to do the sleuthing and answer the questions, we can make do with a much simpler technique, which will allow us to *instantly* get the value of h(n) for a range of n values large enough for our purposes, and does not require recursion at all so it can be easily implemented in most any HP calculator this side of the HP-67 or so. Instead of recursion, we'll use a suitably dimensioned array (or range of registers) to keep the function values, which will be constantly reused to generate further values, substituting recursion for *direct array lookup*, like this:

```
10 DESTROY ALL @ OPTION BASE 1 @ INTEGER H(1000)
20 FOR N=1 TO 1000 @ IF N=1 OR N=3 THEN H(N)=N @ GOTO 70
30 ON RMD(N,4)+1 GOTO 40,50,40,60
40 H(N)=H(N/2) @ GOTO 70
50 H(N)=2*H((N+1)/2)-H((N-1)/4) @ GOTO 70
60 H(N)=3*H((N-1)/2)-2*H((N-3)/4) @ GOTO 70
70 NEXT N @ DISP "Ready: Use FNH(n) (1 <= n <= 1000)" @ END
80 DEF FNH(N)=H(N)
```

Notice just how closely does this mimic the recursive version, only using H(n) (the array) instead of FNH(n) (the recursive call). This version does pre-compute h(1) to h(1000) and stores the results in array H, to be used by the user-defined function FNH(n), just like in the recursive version. Actually, you could use just H(n) directly, instead of FNH(n), but this way both versions work alike and the following examples using them are the same. Don't forget to RUN this version for it to generate the values (RUN is not needed for the recursive defininition).

This makeshift, non-recursive version generates all 1000 values very fast, and using FNH afterwards is much, much faster; however, it does take more memory (to store the values) and is limited to the range 1-1000, while the recursive version doesn't have any such limitation. But for quickly and effortlessly implementing a recursive definition in a hurry, this array technique does deliver the goods !

Second part: The Questions

Using our newly implemented FNH, we can do some sleuthing to try and answer the questions, namely:

1. *"What's the value of h(2^n) ? ditto h(2^n + 1) ? h(2^n - 1) ? h(m * 2^n) ?"*

   For the first question, we will try this, from the command line:

   ```
   FOR N=1 TO 5 @ X=2^N @ DISP X,FNH(X) @ NEXT N
   ```

   You'll get all 1's, so it seems that h(2^n) = 1.

Similar command lines will quickly reveal that h(2^n+1) = 2^n+1 and h(2^n-1) = 2^n-1. As for h(m*2^n), sampling a few values like this:

```
FOR M=14 TO 18 @ FOR N=1 TO 3 @ X=M*2^N @ DISP M;FNH(M);X;FNH(X) @ NEXT N @ NEXT M
```

will make it crystal-clear that h(m*2^n) = h(m).

2. *"For which n is h(n) = 100 ?"*

   Sampling any random values or range of values makes it obvious that h(n) in always odd, so it can never be an even value like 100.

3. *"For which n is h(n) even ?"*

   Ditto. h(n) is always odd.

4. *"If n is odd and m = h(n), what's the value of h(m) ?"*

   Sampling a random range of odd n values, like this:

   ```
   FOR N=431 TO 451 STEP 2 @ DISP N;FNH(FNH(N)) @ NEXT N
   ```

   makes it absolutely clear that in this case h(m) = n.

5. *"What are the values of h(h(n)) ?"*

   Ditto. As we've seen in the previous question, if n is odd then h(h(n)) = n. If n is even, then h(h(n)) is n with all factors 2 casted out.

6. *"For which values of n do we get h(n) = n ?"*

   This one is tricky. As we've seen in question (1) above, if n is a power of two plus or minus one, then h(n)=n. But are there any other values apart from these ? Let's try, add these lines to the program:

   ```
   100 FOR N=100 TO 300 @ IF FNH(N)=N THEN DISP N;
   110 NEXT N @ DISP
   ```

   Now running it (using RUN 100, to avoid clearing the array in the case of the non-recursive version), produces at once:

   ```
   107  119  127  129  153  165  189  195  219  231  255  257  273  297
   ```

   where the powers of 2 plus/minus one do appear (127-129, 255-257), plus a host of other values, but though it's evident that there's some *structure* to it all, it isn't clear what it might be.

However, there are some *signs* here and there waiting for us to notice. Though the function's definition doesn't seem to have much to do with powers of 2, we've found that such arguments are indeed *special*. Exact powers of 2 are always smashed to 1, while their predecessors and successors are returned *intact*. Might it be the case that working in base-10 is actually confusing the issue ? How would it all look *if we used base-2* instead, where powers of 2 are very 'round' numbers (10,100,1000,10000, ...) ?

Let's try ! We'll show both arguments and results in base-2 and see how it all looks like. Altering line 100 above like this:

```
100 FOR N=100 TO 300 @ IF FNH(N)=N THEN DISP N;BSTR$(N,2)
```

and running it again (using RUN 100), we get these results:

```
107   1101011
119   1110111
127   1111111
129   10000001
153   10011001
165   10100101
189   10111101
195   11000011
219   11011011
231   11100111
255   11111111
257   100000001
273   100010001
297   100101001
```

where it is obvious that all n being returned intact by h(n) share a common trait: **their binary representations are palindromic !** Reversing their bit-order leaves them unchanged, they are the mirror images of themselves.

Second part encore: The Big Question [tm]

"Just what on Earth does h(n) do to n ??"

Well, once we've empirically explored how h(n) works and with our intuition sharpened by our previous findings, it's only natural for us to make a bold final statement:

Given an argument n, h(n) massages it a little while and then simply, in three words, **reverses its bits**.

As further confirmation, this command line:

```
FOR N=121 TO 137 STEP 2 @ DISP N,BSTR$(N,2),FNH(N),BSTR$(FNH(N),2)  @ NEXT N
```

produces:

```
  n     n (base2)    h(n)   h(n) (base2)
-----------------------------------------
 121     1111001      79      1001111
 123     1111011     111      1101111
 125     1111101      95      1011111
 127     1111111     127      1111111
 129    10000001     129     10000001
 131    10000011     193     11000001
 133    10000101     161     10100001
 135    10000111     225     11100001
 137    10001001     145     10010001
```

which clearly shows the bit-reversal at work for odd n. Even n are treated likewise, but as they end in one or more zeroes, these are rendered non-significant upon reversal, and do not show up in the output, for instance:

```
n = 136 = 10001000 -> h(n) = 00010001 = 10001 = 17
```

Conclusion

Of course now that we fully know what h(n) does, we can then proceed to write a new, non-recursive, non-array implementation of h(n), like this:

```
10 DEF FNF(N)=BVAL(REV$(BSTR$(N,2)),2)
```

That's all there's to it. This single-line (more like *half-line*) user-defined function will do the same as our previous recursive implementation (reversing the binary bits of its argument), only orders of magnitude faster (a mere hundredths of a second for any legal argument), and using orders of magnitude less memory (neither recursion nor arrays involved, though it does use BVAL and BSTR$, which are keywords from the Math ROM, and REV$ which is a very popular keyword featured in a number of LEX files).

Well, that's as far as it gets. I hope you've found all this interesting and enjoyable and perhaps even enlightening. Whatever the case, I'm sure you'll agree with me that having such a perfecty normal-looking recursive function definition actually perform a *bit-reversal* of its argument is quite *unexpected* and kind of surprising, to say the least !

As I see it, it is like this function, instead of changing the *arithmetic* value of its argument, it rather does alter its *shape* instead, its physical orientation in some sense, disregarding the actual value being involved.

Mathematics are so *beautiful*. And these wonderful HP machines can serve us well to sharpen our insights, our intuition, in this amazing journey.

As for the 'prize', *all of you* who have posted your ideas, programs and insights to this thread and wish to receive a copy of the 1280 x 1024 image in the mail, please send me an e-mail to my e-mail address listed here and I'll send it back to you at once.

Best regards from V.

Edited: 8 July 2004, 10:12 a.m.

I think, I missed the right way...

*Message #23 Posted by **Tizedes Csaba [Hungary]** on 9 July 2004, 2:43 a.m.,*
in response to message #22 by Valentin Albillo

...but I try to find the series without recursion.

Cs.

Re: S&SMC #7: My Solutions And Comments [LOOOOONG]

*Message #24 Posted by **Martin Cohen** on 9 July 2004, 2:50 p.m.,*
in response to message #22 by Valentin Albillo

This is a very cute recursion! How did you come up with it?

Also, you might submit it to the online encyclopedia of integer sequences (if it is not already there).

Martin Cohen

Re: S&SMC #7: My Solutions And Comments [LOOOOONG]

*Message #25 Posted by **Valentin Albillo** on 12 July 2004, 5:34 a.m.,*
in response to message #24 by Martin Cohen

Hi, Martin:

Martin posted:

"This is a very cute recursion! How did you come up with it?"

As they say, "magicians never tell"

"you might submit it to the online encyclopedia of integer sequences (if it is not already there)."

It is: Sequence A030101: Base 2 reversal of N (written in base 10)

```
ID Number: A030101
URL:       http://www.research.att.com/projects/OEIS?Anum=A030101
Sequence:  0,1,1,3,1,5,3,7,1,9,5,13,3,11,7,15,1,17,9,25,5,21,13,29,3,
           19,11,27,7,23,15,31,1,33,17,49,9,41,25,57,5,37,21,53,13,45,
           29,61,3,35,19,51,11,43,27,59,7,39,23,55,15,47,31,63,1,65,33,
           97,17,81,49,113,9,73,41,105,25,89,57
Name:      Base 2 reversal of n (written in base 10).
References Solutions to 17th USA Mat. Olympiad, Math. Mag., 62 (1989), 210-214
              (#3).
Links:     Michael Gilleland, Some Self-Similar Integer Sequences
Formula:   a(n) = 0, a(2n) = a(n), a(2n+1) = a(n) + 2^([log2(n)]+1). For n>0,
              a(n) = 2*A030109(n) - 1. - Ralf Stephan (ralf(AT)ark.in-berlin.de),
              Sep 15 2003
Program:   (PARI) a(n)=if(n<1,0,subst(Polrev(binary(n)),x,2))
See also:  Cf. A030109, A036044, A056539.
           Sequence in context: A000265 A040026 A093474 this_sequence A081432
              A060819 A089654
           Adjacent sequences: A030098 A030099 A030100 this_sequence A030102
              A030103 A030104
```

Thanks for your interest and best regards from V.

Re: S&SMC #7: My Solutions And Comments [LOOOOONG]

*Message #26 Posted by **Carl Nelson** on 10 July 2004, 12:41 a.m.,*
in response to message #22 by Valentin Albillo

Here's a recursive solution for the HP-48G, probably also works on the derivative models as well. YMMV.

Note that the name of the object must match the references in the program. I had made a copy to make some modifications to, and missed one of the references, so I had a little co-recursion happening between two different objects. The amazing thing is that it worked. Well, until I deleted the older version. *Splat*

Also, I am curious here. Has anyone looked at how difficult this would be to implement as a recursive algorithm on a non-stack calculator? i.e. one of the TI calculators, not a higher-language beastie like the HP-71 and variants, that is.

In any case, here is my take on the function.

HP-48G recursive:

HOFN:

<< CASE DUP 1 == THEN END DUP 3 == THEN END DUP 4 MOD CASE DUP 1 == THEN - 4 / DUP 2 * 1 + HOFN 2 * SWAP HOFN - END DUP 3 == THEN - 4 / DUP 2 * 1 + HOFN 3 * SWAP HOFN 2 * - END DROP 2 / HOFN END END >>

Re: S&SMC #7: My Solutions And Comments [LOOOOONG]

*Message #27 Posted by **Carl Nelson** on 10 July 2004, 12:56 a.m.,*
in response to message #26 by Carl Nelson

Let me try that again, with some spacing...

HP-48G recursive:

HOFN:

<< CASE

DUP 1 == THEN END

DUP 3 == THEN END

DUP 4 MOD

CASE

DUP 1 ==

THEN

- 4 / DUP

2 * 1 + HOFN 2 *

SWAP HOFN -

END

DUP 3 ==

THEN

```
- 4 / DUP

2 * 1 + HOFN 3 *

SWAP HOFN 2 * -

END

DROP 2 / HOFN

END

END

>>
```

[Return to Index | Top of Index]

Go back to the main exhibit hall