

Strings of Consecutive Happy Numbers

23 Feb 2008

1 Mar 2008

15 Mar 2008

21 Mar 2008

29 June 2009

The goal here is to show that the first string of ten consecutive happy numbers.

Note that Dr. Grundman uses S_2 but we will simply use S in the explanation, which in our program is the procedure 'onestep'.

Also note we will use a dot as the digit concatenation operator. Thus, 111112999994 could be written 11111.2.9999.94 if we wish.

First we define our procedures.

```
> restart;
> f := n → n^2; #in case we ever want to investigate the cube of the digits, etc.
                                     f := n → n2 (1)
```

```
> bs := 10;
    #this is the base, in case we ever want to investigate binary or ternary or any other base.
                                     bs := 10 (2)
```

```
> onestep := proc(n1)
    #this is what Dr. Grundman calls S_2(n1) and what we will simply call S below.
    local ans, n, d;
    n := n1;
    ans := 0;
    while n > 0 do
    d := n mod bs;
    ans := ans + f(d);
    n := (n - d) / bs;
    end do;
    ans;
end;
```

```
onestep := proc(n1) (3)
    local ans, n, d;
    n := n1;
    ans := 0;
    while 0 < n do d := mod(n, bs); ans := ans + f(d); n := (n - d) / bs end do;
    ans
end proc
```

```
> happy := proc(n)
    # returns -1 if not happy, and returns the number of steps to reach 1 if it is happy
    local m, j, height;
    m := n;
```

```

height := -1;
for j from 1 to 100 while (m > 1 and m ≠ 4) do
m := onestep(m);
end do;
if m = 1 then height := j; end if;
height;
end;

```

happy := proc(n)

local m, j, height;

m := n;

height := -1;

for j to 100 while 1 < m and m <> 4 do m := onestep(m) end do;

if m = 1 then height := j end if;

height

end proc

(4)

>

The next procedure is only needed when we want to find the smallest N with $S(N) = n$ for a given n. A separate worksheet has the details on how this is constructed. The array contains the smallest N for $1 \leq n \leq 486 = 6 \cdot 81$.

>

```

lowS := [1, 11, 111, 2, 12, 112, 1112, 22, 3, 13, 113, 222, 23, 123, 1123, 4, 14, 33, 133, 24,
124, 233, 1233, 224, 5, 15, 115, 1115, 25, 125, 1125, 44, 144, 35, 135, 6, 16, 116, 1116,
26, 45, 145, 335, 226, 36, 136, 1136, 444, 7, 17, 117, 46, 27, 127, 1127, 246, 227, 37, 137,
1137, 56, 156, 1156, 8, 18, 118, 337, 28, 128, 356, 1356, 66, 38, 57, 157, 266, 238, 257,
1257, 48, 9, 19, 119, 248, 29, 129, 1129, 466, 58, 39, 139, 1139, 258, 239, 1239, 448, 49,
77, 177, 68, 168, 277, 1277, 268, 458, 59, 159, 666, 368, 259, 1259, 2666, 78, 178, 359,
468, 69, 169, 1169, 2468, 269, 378, 577, 1577, 568, 369, 1369, 88, 188, 79, 179, 288, 469,
279, 1279, 668, 388, 578, 379, 1379, 2388, 569, 1569, 488, 89, 189, 777, 1777, 289, 1289,
2777, 4668, 588, 389, 579, 1579, 2588, 2389, 2579, 4488, 489, 99, 199, 688, 1688, 299,
1299, 2688, 4588, 589, 399, 1399, 3688, 2589, 2399, 12399, 788, 499, 779, 1779, 689,
1689, 2779, 12779, 2689, 3788, 599, 1599, 5688, 3689, 2599, 888, 1888, 789, 1789, 2888,
4689, 699, 1699, 6688, 3888, 2699, 3789, 5779, 15779, 5689, 3699, 4888, 889, 1889, 799,
1799, 2889, 4699, 2799, 12799, 5888, 3889, 5789, 3799, 13799, 23889, 5699, 15699,
4889, 899, 1899, 6888, 16888, 2899, 12899, 26888, 45888, 5889, 3899, 5799, 15799,
25889, 23899, 25799, 7888, 4899, 999, 1999, 6889, 16889, 2999, 12999, 26889, 37888,
5899, 3999, 13999, 36889, 25899, 8888, 18888, 7889, 4999, 7799, 17799, 6899, 16899,
27799, 38888, 26899, 37889, 5999, 15999, 56889, 36899, 25999, 8889, 18889, 7899,
17899, 28889, 46899, 6999, 16999, 58888, 38889, 26999, 37899, 57799, 157799, 56899,
36999, 48889, 8899, 18899, 7999, 17999, 28899, 46999, 27999, 127999, 58889, 38899,
57899, 37999, 137999, 238899, 56999, 78888, 48899, 8999, 18999, 68889, 168889,
28999, 128999, 268889, 378888, 58899, 38999, 57999, 157999, 258899, 88888, 188888,
78889, 48999, 9999, 19999, 68899, 168899, 29999, 129999, 268899, 378889, 58999,
39999, 139999, 368899, 258999, 88889, 188889, 78899, 49999, 77999, 177999, 68999,
168999, 277999, 388889, 268999, 378899, 59999, 159999, 568899, 368999, 259999,
88899, 188899, 78999, 178999, 288899, 468999, 69999, 169999, 588889, 388899,
269999, 378999, 577999, 1577999, 568999, 369999, 488899, 88999, 188999, 79999,
179999, 288999, 469999, 279999, 1279999, 588899, 388999, 578999, 379999, 1379999,
888888, 569999, 788889, 488999, 89999, 189999, 688899, 1688899, 289999, 1289999,
2688899, 3788889, 588999, 389999, 579999, 1579999, 2588999, 888889, 1888889,
788899, 489999, 99999, 199999, 688999, 1688999, 299999, 1299999, 2688999, 3788899,

```



```

1, 7
2, 8
3, 5
4, 7
5, 6
6, 7
7, 5
8, 6
9, 6
10, -1

```

(7)

> onestep(N + 4)

```
52561
```

(8)

We check for five or more in a row before the carry, then 5 or more after. Note that these calculations show there are never six in a row on either side of the carry.

>

> for a from 1 to 60000 do

```

  if happy(a + onestep(9)) > 0 and happy(a + onestep(8)) > 0 and happy(a + onestep(7))
    > 0 and happy(a + onestep(6)) > 0 and happy(a + onestep(5)) > 0 then print(a,
    happy(a + onestep(4)), happy(a + onestep(3))); end if;

```

end do:

```

12680, -1, -1
17180, -1, -1
21680, -1, -1
25480, -1, 6
27080, -1, -1
33580, -1, 6
36280, -1, -1
52480, -1, 6
55180, -1, -1

```

(9)

> for a from 1 to 60000 do

```

  if happy(a + onestep(0)) > 0 and happy(a + onestep(1)) > 0 and happy(a + onestep(2))
    > 0 and happy(a + onestep(3)) > 0 and happy(a + onestep(4)) > 0 then print(a,
    happy(a + onestep(5)), happy(a + onestep(6))); end if;

```

end do:

```

11834, -1, -1
14734, -1, -1
17434, -1, -1
18134, -1, -1
22734, -1, -1
27234, -1, -1
41734, -1, -1
44534, -1, -1
45434, -1, -1
45534, -1, -1

```

```

47134, -1, -1
54434, -1, -1
54534, -1, -1
55434, -1, -1

```

(10)

We never have six in a row so must have 5,6,7,8,9 then carry to 0,1,2,3,4.

```

> a := 12680;
  ktop := iquo(a, 81);
  for k from 0 to ktop do # k is the number of nines at the end of N1.
  for d from 0 to 8 do # d is the digit that precedes the k digits of nine at the end of N1.
  if happy(a + 2·d + 1 - k·81) > 0 and happy(a + 2·d + 1 - k·81 + onestep(1)) > 0
    and happy(a + 2·d + 1 - k·81 + onestep(2)) > 0 and happy(a + 2·d + 1 - k·81
      + onestep(3)) > 0 then print(a, k, d, happy(a + 2·d + 1 - k·81 + onestep(4)), a - d2
        - k·81) end if;
  end do; end do;

```

(11)

```

a := 12680
ktop := 156

```

```

> a := 17180;
  ktop := iquo(a, 81);
  for k from 0 to ktop do # k is the number of nines at the end of N1.
  for d from 0 to 8 do # d is the digit that precedes the k digits of nine at the end of N1.
  if happy(a + 2·d + 1 - k·81) > 0 and happy(a + 2·d + 1 - k·81 + onestep(1)) > 0
    and happy(a + 2·d + 1 - k·81 + onestep(2)) > 0 and happy(a + 2·d + 1 - k·81
      + onestep(3)) > 0 then print(a, k, d, happy(a + 2·d + 1 - k·81 + onestep(4)), a - d2
        - k·81) end if;
  end do; end do;

```

(12)

```

a := 17180
ktop := 212
17180, 74, 7, -1, 11137

```

```

> a := 21680;
  ktop := iquo(a, 81);
  for k from 0 to ktop do # k is the number of nines at the end of N1.
  for d from 0 to 8 do # d is the digit that precedes the k digits of nine at the end of N1.
  if happy(a + 2·d + 1 - k·81) > 0 and happy(a + 2·d + 1 - k·81 + onestep(1)) > 0
    and happy(a + 2·d + 1 - k·81 + onestep(2)) > 0 and happy(a + 2·d + 1 - k·81
      + onestep(3)) > 0 then print(a, k, d, happy(a + 2·d + 1 - k·81 + onestep(4)), a - d2
        - k·81) end if;
  end do; end do;

```

(13)

```

a := 21680
ktop := 267
21680, 25, 5, -1, 19630
21680, 67, 0, -1, 16253
21680, 88, 4, -1, 14536

```

```

> a := 25480;
  ktop := iquo(a, 81);
  for k from 0 to ktop do # k is the number of nines at the end of N1.
  for d from 0 to 8 do # d is the digit that precedes the k digits of nine at the end of N1.
  if happy(a + 2·d + 1 - k·81) > 0 and happy(a + 2·d + 1 - k·81 + onestep(1)) > 0

```

```

and happy(a + 2·d + 1 - k·81 + onestep(2)) > 0 and happy(a + 2·d + 1 - k·81
+ onestep(3)) > 0 then print(a, k, d, happy(a + 2·d + 1 - k·81 + onestep(4)), a - d2
- k·81) end if;
end do; end do:

```

a := 25480

k_{top} := 314

(14)

```

> a := 27080;
  ktop := iquo(a, 81);
  for k from 0 to ktop do # k is the number of nines at the end of N1.
  for d from 0 to 8 do # d is the digit that precedes the k digits of nine at the end of N1.
  if happy(a + 2·d + 1 - k·81) > 0 and happy(a + 2·d + 1 - k·81 + onestep(1)) > 0
  and happy(a + 2·d + 1 - k·81 + onestep(2)) > 0 and happy(a + 2·d + 1 - k·81
+ onestep(3)) > 0 then print(a, k, d, happy(a + 2·d + 1 - k·81 + onestep(4)), a - d2
- k·81) end if;
end do; end do:

```

a := 27080

k_{top} := 334

27080, 67, 0, -1, 21653

27080, 74, 7, -1, 21037

27080, 125, 5, -1, 16930

27080, 267, 0, -1, 5453

(15)

```

> a := 33580;
  ktop := iquo(a, 81);
  for k from 0 to ktop do # k is the number of nines at the end of N1.
  for d from 0 to 8 do # d is the digit that precedes the k digits of nine at the end of N1.
  if happy(a + 2·d + 1 - k·81) > 0 and happy(a + 2·d + 1 - k·81 + onestep(1)) > 0
  and happy(a + 2·d + 1 - k·81 + onestep(2)) > 0 and happy(a + 2·d + 1 - k·81
+ onestep(3)) > 0 then print(a, k, d, happy(a + 2·d + 1 - k·81 + onestep(4)), a - d2
- k·81) end if;
end do; end do:

```

a := 33580

k_{top} := 414

(16)

```

> a := 36280;
  ktop := iquo(a, 81);
  for k from 0 to ktop do # k is the number of nines at the end of N1.
  for d from 0 to 8 do # d is the digit that precedes the k digits of nine at the end of N1.
  if happy(a + 2·d + 1 - k·81) > 0 and happy(a + 2·d + 1 - k·81 + onestep(1)) > 0
  and happy(a + 2·d + 1 - k·81 + onestep(2)) > 0 and happy(a + 2·d + 1 - k·81
+ onestep(3)) > 0 then print(a, k, d, happy(a + 2·d + 1 - k·81 + onestep(4)), a - d2
- k·81) end if;
end do; end do:

```

a := 36280

k_{top} := 447

(17)

```

> a := 52480;
  ktop := iquo(a, 81);
  for k from 0 to ktop do # k is the number of nines at the end of N1.
  for d from 0 to 8 do # d is the digit that precedes the k digits of nine at the end of N1.

```


