

Strings of Consecutive Happy Numbers

23 Feb 2008

1 Mar 2008

15 Mar 2008

21 Mar 2008

1 Sept 2008

6 Sept 2008

10 Sept 2008

12 Sept 2008

18 Sept 2008

19 Sept 2008

26 Sept 2008

9 Oct 2008

22 Jan 2009

5 Feb 2009

27 June 2009

24 July 2009

23 Sept 2009

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

The 23319753086419750617286 digit number

$N = 1.(812181234431825799737 \text{ nines}).5.(22507571851987924817546 \text{ nines}).2$

is the best example we found so far via a random search, and has the right form with the correct number of digits, but is probably not the best.

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.

```
> 2 + 2222222222222220 + 1 + 97388 + 1; M1 := onestep(77) + 2222222222222220·81 + 32
    + 97388·81;
```

```
                2222222222319612
                M1 := 180000000007888355 (1)
```

```
> restart;
```

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

```
                f := n → n2 (2)
```

```
> bs := 10;
```

```
    #this is the base, in case we ever want to investigate binary or ternary or any other base.
```

```
                bs := 10 (3)
```

```
> 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) (4)
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) (5)
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

```

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, 156999, 48899, 8999, 18999, 68889, 168889, 28999, 128999, 268889, 458889, 58899, 38999, 57999, 157999, 258889, 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, 488889, 88999, 188999, 79999, 179999, 288899, 469999, 279999, 1279999, 588899, 388999, 578999, 379999, 1379999, 2388899, 569999, 1569999, 488899, 89999, 188999, 688899, 1688899, 289999, 1289999, 2688899, 3788889, 588999, 389999, 579999, 1579999, 2588999, 888889, 1888889, 788899, 489999, 99999, 199999, 688999, 1688999, 299999, 1299999, 2688999, 3788899, 589999, 399999, 1399999, 3688999, 2589999, 888899, 1888899, 788999, 499999, 779999, 1779999, 689999, 1689999, 2779999, 3888899, 2689999, 3788999, 599999, 1599999, 5688999, 3689999, 2599999, 888999, 1888999, 789999, 1789999, 2888999, 4689999, 699999, 1699999, 5888899, 3888999, 2699999, 3789999, 5779999, 15779999, 5689999, 3699999, 4888999, 889999, 1889999, 799999, 1799999, 2889999, 4699999, 2799999, 12799999, 5888999, 3889999, 5789999, 3799999, 13799999, 2388889, 5699999, 15699999, 4888899, 8899999, 1888899, 6888899, 16888899, 2899999, 12899999, 26888899, 3788889, 5889999, 3899999, 5799999, 15799999, 25889999, 8888899, 18888899, 7888999, 4899999, 9999999] :

```
> minimalN := proc(n)
  local q, r, k, ans;
  global lowS;;
  if n < 487 then ans := lowS[n];
  else
    q := iquo(n, 81, 'r');
    ans := lowS[n - (q - 5) · 81] · 10q-5 + (10q-5 - 1);
  end if;
  ans;
end;
```

```
minimalN := proc(n)
  local q, r, k, ans;
  global lowS;
  if n < 487 then
    ans := lowS[n]
  else
    q := iquo(n, 81, 'r'); ans := lowS[n - 81 * q + 405] * 10^(q - 5) + 10^(q - 5) - 1
```

(6)

```

end if;
ans
end proc
> minimalbigN := proc(n)
  local q, r, k, ans;
  global lowS;;
  if n < 487 then ans := lowS[n];
  else
  if n < 1000 then
q := iquo(n, 81, 'r');
ans := lowS[n - (q - 5) · 81] · 10q-5 + (10q-5 - 1);
  else
q := iquo(n, 81, 'r');
ans := [ lowS[n - (q - 5) · 81], q - 5];
  end if; end if;
ans;
end;

```

```
minimalbigN := proc(n)
```

(7)

```

  local q, r, k, ans;
  global lowS;
  if n < 487 then
    ans := lowS[n]
  else
    if n < 1000 then
      q := iquo(n, 81, 'r'); ans := lowS[n - 81 * q + 405] * 10(q - 5) + 10
      ^ (q - 5) - 1
    else
      q := iquo(n, 81, 'r'); ans := [lowS[n - 81 * q + 405], q - 5]
    end if
  end if;
  ans
end proc

```

We try to find an 8/8 split over the carry.

The idea is that $N = N1.d0$ where $d0$ is the final digit. Let $M1 = S(N1)$ and we divide $M1$ into the last three digits and the preceding digits, call them $M2.c$ where c is the final three digits.

If $c+d0^2 < 1000$ then $S(S(N)) = S(M2) + S(c+d0^2)$ so we can merely check this to see if it is happy or not. Below we use $b=S(M2)$ and $d0 = i$ for the various final digits we are trying.

Recall that to guarantee that $c+d0^2 < 1000$ we investigate $c < 919 = 1000 - 9^2$ for those "before the carry" and $c < 1000 - 7^2 = 951$ for those "after the carry."

First we search for 8 in a row before the carry.

```

> for b from 0 to 2000 do
  for c from 0 to 918 do

```


for c from 984 to 990 do #so these have the last digit 9,8,7,6,5,4 carry but not any others.

fini := false;

for i from 2 to 3 while *fini* = false **do**

if happy($b + \text{onestep}(c + i^2)$) = -1 **then** *fini* := true; **end if;**

end do;

if *fini* = false **then** #now we check for last digit 9,8,7,6,5,4

for k from 0 to floor($\frac{b}{81}$) **do**

for d2 from 0 to 8 do

fini2 := false;

for i from 4 to 9 while *fini2* = false **do**

if happy($b + 2 \cdot d2 + 1 - k \cdot 81 + \text{onestep}((c + i^2) \bmod 100)$) = -1 **then** *fini2* := true; **end if;**

end do;

if *fini2* = false **then** print($b, c, k, d2, \text{happy}(b + \text{onestep}(c + 1^2)), \text{happy}(b + \text{onestep}(c)),$

$\text{minimalN}(b - d2^2 - k \cdot 81) \cdot 10^{(k+4)} + d2 \cdot 10^{k+3} + 10^{k+3} - 1000 + c,$

$\text{evalf}(\log_{10}(\text{minimalN}(b - d2^2 - k \cdot 81) \cdot 10^{(k+4)} + d2 \cdot 10^{k+3} + 10^{k+3} - 1000 + c))$)

end if;

end do;

end do;

end if;

end do:

end do:

> **for b from 0 to 2000 do**

for c from 991 to 995 do #so these have the last digit 9,8,7,6,5,4,3 carry but not any others.

fini := false;

for i from 2 to 2 while *fini* = false **do**

if happy($b + \text{onestep}(c + i^2)$) = -1 **then** *fini* := true; **end if;**

end do;

if *fini* = false **then** #now we check for last digit 9,8,7,6,5,4,3

for k from 0 to floor($\frac{b}{81}$) **do**

for d2 from 0 to 8 do

fini2 := false;

for i from 3 to 9 while *fini2* = false **do**

if happy($b + 2 \cdot d2 + 1 - k \cdot 81 + \text{onestep}((c + i^2) \bmod 100)$) = -1 **then** *fini2* := true; **end if;**

end do;

if *fini2* = false **then** print($b, c, k, d2, \text{happy}(b + \text{onestep}(c + 1^2)), \text{happy}(b + \text{onestep}(c)),$

$\text{minimalN}(b - d2^2 - k \cdot 81) \cdot 10^{(k+4)} + d2 \cdot 10^{k+3} + 10^{k+3} - 1000 + c,$

$\text{evalf}(\log_{10}(\text{minimalN}(b - d2^2 - k \cdot 81) \cdot 10^{(k+4)} + d2 \cdot 10^{k+3} + 10^{k+3} - 1000 + c))$)

end if;

end do;

end do;

end if;

end do:

end do:

> **for b from 0 to 2000 do**

for c from 996 to 999 do #so these have the last digit 9,8,7,6,5,4,3,2 carry.

```

#fini :=false;
#for i from 3 to 4 while fini = false do
#if happy(b + onestep(c + i2)) = -1 then fini := true; end if;
#end do;
#if fini=false then #now we check for last digit 9,8,7,6,5,4,3,2
  for k from 0 to floor( $\frac{b}{81}$ ) do
    for d2 from 0 to 8 do
      fini2 := false;
      for i from 2 to 9 while fini2 = false do
        if happy(b + 2·d2 + 1 - k·81 + onestep((c + i2) mod 100)) = -1 then fini2 := true; end
          if;
        end do;
      if fini2 = false then print(b, c, k, d2, happy(b + onestep(c + 12)), happy(b + onestep(c)),
        minimalN(b - d22 - k·81) · 10(k+4) + d2·10k+3 + 10k+3 - 1000 + c,
        evalf(log10(minimalN(b - d22 - k·81) · 10(k+4) + d2·10k+3 + 10k+3 - 1000 + c)))
        end if;
      end do;
    end do;
  #end if;
end do;
end do;

```

```

> evalf(log10(minimalN(1710)·1000 + 534));
24.60205999 (13)

```

So the candidate values of b are 1710, 1654, 1602, 1647, 1654, and 1663. The smallest M1 value resulting from these is the 25 digit number
M1 = 1888899999999999799999949
corresponding to (b,c,k,d2) = (1602, 949, 5, 7) which has precisely eight consecutive happy numbers before the carry.

We see here that the digit d=5 will lead to the smallest potential N.

```

> a := 1888899999999999799999949;
  for d from 0 to 8 do print(d, minimalbigN(a - d2) end do;
    a := 1888899999999999799999949
    0, [1599999, 23319753086419750617278]
    1, [599999, 23319753086419750617278]
    2, [3888899, 23319753086419750617278]
    3, [779999, 23319753086419750617278]
    4, [1399999, 23319753086419750617278]
    5, [199999, 23319753086419750617278]
    6, [37888899, 23319753086419750617277]
    7, [3799999, 23319753086419750617277]
    8, [8888888, 23319753086419750617277] (14)

```

```

> a := 1888899999999999799999949 ;
  ktop := iquo(a, 81);
  for k from ktop to 0 by -1 do # k is the number of nines at the end of N1.

```



```

for d2 from 0 to 8 do
  fini2 := false;
  for i from 6 to 7 while fini2 = false do
    if happy(b + 2·d2 + 1 - k·81 + onestep((c + i2) mod 100)) = -1 then fini2 := true; end
    if;
  end do;
  if fini2 = false then print(b, c, k, d2, happy(b + 2·d2 + 1 - k·81 + onestep((c + 82)
    mod 100))) end if;
  end do;
end do;
end if;
end do;
end do;

```

```

> for b from 0 to 2000 do
  for c from 975 to 983 do #so these have the last digit 5 carry but not any others.
  fini := false;
  for i from 0 to 4 while fini = false do
    if happy(b + onestep(c + i2)) = -1 then fini := true; end if;
  end do;
  if fini = false then #now we check for last digit 5
    for k from 0 to floor( $\frac{b}{81}$ ) do
      for d2 from 0 to 8 do #d2 is the digit of the b part that precedes the nines that carry.
      fini2 := false;
      for i from 5 to 7 while fini2 = false do
        if happy(b + 2·d2 + 1 - k·81 + onestep((c + i2) mod 100)) = -1 then fini2 := true; end
        if;
      end do;
      if fini2 = false then print(b, c, k, d2, happy(b + 2·d2 + 1 - k·81 + onestep((c + 82)
        mod 100))) end if;
      end do;
    end do;
  end do;
end do;

```

1663, 978, 7, 4, 4

(17)

```

> for b from 0 to 2000 do
  for c from 984 to 990 do #so these have the last digit 5 and 4 carry but not any others.
  fini := false;
  for i from 0 to 3 while fini = false do
    if happy(b + onestep(c + i2)) = -1 then fini := true; end if;
  end do;
  if fini = false then #now we check for last digit 5,4
    for k from 0 to floor( $\frac{b}{81}$ ) do
      for d2 from 0 to 8 do
      fini2 := false;
      for i from 4 to 7 while fini2 = false do
        if happy(b + 2·d2 + 1 - k·81 + onestep((c + i2) mod 100)) = -1 then fini2 := true; end
        if;
      end do;
    end do;
  end do;

```

```

    if;
  end do;
  if fini2 = false then print(b, c, k, d2, happy(b + 2·d2 + 1 - k·81 + onestep((c + 82)
    mod 100))) end if;
  end do;
end do;
end do;
end do;
end do;
end do;

```

```

> for b from 0 to 2000 do
  for c from 991 to 995 do #so these have the last digit 5,4,3 carry but not any others.
    fini := false;
    for i from 0 to 2 while fini = false do
      if happy(b + onestep(c + i2)) = -1 then fini := true; end if;
    end do;
    if fini = false then #now we check for last digit 5,4,3
      for k from 0 to floor( $\frac{b}{81}$ ) do
        for d2 from 0 to 8 do
          fini2 := false;
          for i from 3 to 7 while fini2 = false do
            if happy(b + 2·d2 + 1 - k·81 + onestep((c + i2) mod 100)) = -1 then fini2 := true; end
              if;
            end do;
            if fini2 = false then print(b, c, k, d2, happy(b + 2·d2 + 1 - k·81 + onestep((c + 82)
              mod 100))) end if;
            end do;
          end do;
        end if;
      end do;
    end do;
  end do;
end do;

```

1582, 994, 16, 7, -1

(18)

```

> for b from 0 to 2000 do
  for c from 996 to 998 do #so these have the last digit 5,4,3,2 carry but not any others.
    fini := false;
    for i from 0 to 1 while fini = false do
      if happy(b + onestep(c + i2)) = -1 then fini := true; end if;
    end do;
    if fini = false then #now we check for last digit 5,4,3,2
      for k from 0 to floor( $\frac{b}{81}$ ) do
        for d2 from 0 to 8 do
          fini2 := false;
          for i from 2 to 7 while fini2 = false do
            if happy(b + 2·d2 + 1 - k·81 + onestep((c + i2) mod 100)) = -1 then fini2 := true; end
              if;
            end do;
            if fini2 = false then print(b, c, k, d2, happy(b + 2·d2 + 1 - k·81 + onestep((c + 82)
              mod 100))) end if;
            end do;
          end do;
        end if;
      end do;
    end do;
  end do;
end do;

```

```

end do;
end do;
end if;
end do;
end do:

```

```

> for b from 0 to 1500 do
  for c from 999 to 999 do #so these have the last digit 5,4,3,2,1 carry but not any others.
    fini := false;
    for i from 0 to 0 while fini = false do
      if happy(b + onestep(c + i^2)) = -1 then fini := true; end if;
    end do;
    if fini = false then #now we check for last digit 5
      for k from 0 to floor( (b / 81) ) do
        for d2 from 0 to 8 do
          fini2 := false;
          for i from 1 to 7 while fini2 = false do
            if happy(b + 2·d2 + 1 - k·81 + onestep((c + i^2) mod 100)) = -1 then fini2 := true; end
              if;
            end do;
            if fini2 = false then print(b, c, k, d2, happy(b + 2·d2 + 1 - k·81 + onestep((c + 8^2)
              mod 100))) end if;
            end do;
          end do;
        end if;
      end do;
    end do:

```

1103 leads to the 18 digit M1 = 179999999999999934
 The 1582 leads to the 25 digit M1 = 15799.7.(16 nines).994
 1663 leads to the 26 digit M1 = 5688.(11 nines).4.(7 nines).978

$$\begin{aligned}
 > 1582 - 16 \cdot 81 - 7^2; \text{minimalbigN}(1582 - 16 \cdot 81 - 7^2); \\
 & \qquad \qquad \qquad 237 \\
 & \qquad \qquad \qquad 15799 \qquad \qquad \qquad (19)
 \end{aligned}$$

$$\begin{aligned}
 > 1663 - 7 \cdot 81 - 4^2; \text{minimalbigN}(\%); \\
 & \qquad \qquad \qquad 1080 \\
 & \qquad \qquad \qquad [5688999, 8] \qquad \qquad \qquad (20)
 \end{aligned}$$

We will try to match with the known cases of 8 in a row after the carry. First try to match the 1103 to 1184 allhaving the last two digits are 34.

Then $a+2d+1-k \cdot 81 = 934 \pmod{1000}$, and $d=5$, so $k = 346$.

$$> a := 1888899999999999799999949; \frac{(a + 11 - 34)}{81} \pmod{100};$$

```
a := 1888899999999999799999949
46 (21)
```

```
> a := 1888899999999999799999949;
d := 5; # d is the digit that precedes the k digits of nine at the end of N1.
ktop := iquo(a, 81);
a3 := (ktop - 46) mod 100;
for k from (ktop - a3) to 0 by -1000 do # k is the number of nines at the end of N1.
q := a + 2·d + 1 - k·81;
if onestep(q - 34) = 1184 then print(k, q, onestep(q - 34)) end if;
end do;
print(k, "done");
```

```
a := 1888899999999999799999949
d := 5
ktop := 23319753086419750617283
a3 := 37
```

```
Warning, computation interrupted
23319753084745768112246, "done" (22)
```

```
> k;
23319753084745768112246 (23)
```

Now we try to match the 1582, so the last three digits are 994.
Then $a+2d+1-k\cdot 81 = 994 \pmod{1000}$, and $d=5$, so $k = 86$.

```
> a := 1888899999999999799999949 ; (a + 11 - 994) / 81 mod 1000;
a := 1888899999999999799999949
86 (24)
```

```
> a := 1888899999999999799999949;
d := 5; # d is the digit that precedes the k digits of nine at the end of N1.
ktop := iquo(a, 81);
a3 := (ktop - 86) mod 1000;
for k from (ktop - a3) to 0 by -1000 do # k is the number of nines at the end of N1.
q := a + 2·d + 1 - k·81;
if onestep(q - 994) = 1103 then print(k, q, onestep(q - 994)) end if;
end do;
print(k, "done");
```

```
a := 1888899999999999799999949
d := 5
ktop := 23319753086419750617283
a3 := 197
```

```
Warning, computation interrupted
23319753085828461490086, "done" (25)
```

```
> k;
23319753085828461490086 (26)
```

No luck so far. Let us try finding random values that do work; then at least we know we are not searching in vain, plus then we know we have the correct number of digits for the smallest example of

16 in a row.

The form must be

$N = 1.(23319753086419750617283 - k \text{ nines}).5.(k \text{ nines}).2$ where we need to find a suitable k . This has 23319753086419750617286 *digits*.

We verify this gives eight in a row before the carry.

```
> minimalbigN(18888999999999999999799999949 - 5^2);  
[199999, 23319753086419750617278] (27)
```

```
> a1 := 1 + 23319753086419750617283 * 81 + 5^2; for d from 1 to 9 do print(d, happy(a1  
+ d^2)) end do;
```

```
a1 := 18888999999999999999799999949
```

```
1, -1
```

```
2, 4
```

```
3, 5
```

```
4, 6
```

```
5, 5
```

```
6, 5
```

```
7, 8
```

```
8, 5
```

```
9, 4
```

(28)

```
> a := 18888999999999999999799999949;  
ktop := iquo(a, 81);  
r2 := rand(4..22);  
d := 5;
```

```
for j from 1 to 1000000 do
```

```
  r3 := rand(10^4..4*10^r2());
```

```
  k := r3();
```

```
  # k is the number of nines at the end of N1. log-randomized
```

```
  if k < ktop then
```

```
    # d is the digit that precedes the k digits of nine at the end of N1.
```

```
    q := a + 2*d + 1 - k*81;
```

```
    if happy(q) > 0 and happy(q + 1^2) > 0 and happy(q + 2^2) > 0 and happy(q + 3^2) > 0
```

```
      and happy(q + 4^2) > 0 and happy(q + 5^2) > 0 then print(a, k, d, happy(q + 6^2),
```

```
        happy(q + 7^2), a - d^2 - k*81) end if;
```

```
    end if;
```

```
  end do;
```

```
a := 18888999999999999999799999949
```

```
ktop := 23319753086419750617283
```

```
r2 := proc()
```

```
  proc() option builtin = RandNumberInterface; end proc(6, 19, 5) + 4
```

```
end proc
```

```
d := 5
```

```
18888999999999999999799999949, 136284020887753746, 5, -1, 6, 188888960994307891946498
```

```
18888999999999999999799999949, 3122749145486593672620, 5, -1, 6,
```


$d := 5$

18888999999999997999999949, 2300409733119298833520, 5, -1, 6,
1702566811617336594484804

18888999999999997999999949, 3447355515628697920, 5, -1, 6,
1888620764203233875468404

18888999999999997999999949, 13879188548893733306820, 5, -1, 6,
764685727539607402147504

18888999999999997999999949, 38784898947946399420, 5, -1, 6,
1885758423185216141646904

18888999999999997999999949, 439980311842489288320, 5, -1, 6,
1853261594740758167646004

18888999999999997999999949, 2994131929346409820, 5, -1, 6,
1888657475313722740804504

18888999999999997999999949, 12759660803491679745220, 5, -1, 6,
855367474917173740637104

18888999999999997999999949, 17707097589799264976620, 5, -1, 6,
454625095226259336893704

18888999999999997999999949, 37862644590028725120, 5, -1, 6,
1885833125788207473265204

18888999999999997999999949, 7957842860642664120, 5, -1, 6,
1888255414728287744206204

18888999999999997999999949, 2991018662303956265420, 5, -1, 6,
1646627488353379342500904

18888999999999997999999949, 3756414766220084460920, 5, -1, 6,
1584630403936172958665404

18888999999999997999999949, 6930973573250349820, 5, -1, 6,
1888338591140566521664504

18888999999999997999999949, 171174027820295767820, 5, -1, 6,
1875034903746555842806504

18888999999999997999999949, 154126656149799580920, 5, -1, 6,
1876415740851866033945404

18888999999999997999999949, 12857546405762302820, 5, -1, 6,
1887858538741133053471504

18888999999999997999999949, 288123834341272538220, 5, -1, 6,
1865561969418356724404104

18888999999999997999999949, 1698241565608609520, 5, -1, 6,
1888762442433185502628804

18888999999999997999999949, 2356367585883540964720, 5, -1, 6,
1698034225543432981857604

18888999999999997999999949, 176850874669792447820, 5, -1, 6,
1874575079151746611726504

18888999999999997999999949, 27391039925524476920, 5, -1, 6,


```

for j from 1 to 1000000 do
  r3 := rand(104..4·10r2());
  k := r3()·1000 + 846;
  # k is the number of nines at the end of N1. log-randomized
  if k < ktop then
    # d is the digit that precedes the k digits of nine at the end of N1.
    q := a + 2·d + 1 - k·81;
    if happy(q) > 0 and happy(q + 12) > 0 and happy(q + 22) > 0 and happy(q + 32) > 0
      and happy(q + 42) > 0 and happy(q + 52) > 0 and happy(q + 62) > 0 then print(a,
        k, d, happy(q + 72), happy(q + 82), a - d2 - k·81) end if;
    end if;
  end do;

  a := 18888999999999997999999949
  ktop := 23319753086419750617283

```

```
r2 := proc( )
```

```
  proc( ) option builtin = RandNumberInterface; end proc(6, 16, 4) + 4
```

```
end proc
```

```
  d := 5
```

```

18888999999999997999999949, 384842176846, 5, 6, -1, 1888899999968827583675398
18888999999999997999999949, 142531235823846, 5, 6, -1, 1888899988454969698268398
18888999999999997999999949, 90125188213846, 5, 6, -1, 1888899992699859554678398
18888999999999997999999949, 395497053846, 5, 6, -1, 188889999967964538638398
18888999999999997999999949, 263399634846, 5, 6, -1, 188889999978664429577398
18888999999999997999999949, 392728768444846, 5, 6, -1, 1888899968188969555967398
18888999999999997999999949, 179086789153846, 5, 6, -1, 1888899985493969878538398
18888999999999997999999949, 83370642369382846, 5, 6, -1, 1888893246977967879989398
18888999999999997999999949, 3039054846, 5, 6, -1, 188889999999753636557398
18888999999999997999999949, 39059917846, 5, 6, -1, 188889999996835946654398
18888999999999997999999949, 523763118846, 5, 6, -1, 1888899999957574987373398
18888999999999997999999949, 275755729846, 5, 6, -1, 1888899999977663585882398
18888999999999997999999949, 39912538846, 5, 6, -1, 188889999996766884353398
18888999999999997999999949, 435012407846, 5, 6, -1, 1888899999964763794964398
18888999999999997999999949, 876916434618846, 5, 6, -1, 1888899928969768595873398
18888999999999997999999949, 623462360014846, 5, 6, -1, 1888899949499548638797398
18888999999999997999999949, 2901236127846, 5, 6, -1, 1888899999764999673644398
18888999999999997999999949, 3745715622846, 5, 6, -1, 1888899999696596834549398
18888999999999997999999949, 7942839544846, 5, 6, -1, 1888899999356629796867398
18888999999999997999999949, 17581066846, 5, 6, -1, 188889999998575733585398
18888999999999997999999949, 2890585846, 5, 6, -1, 188889999999765662546398
18888999999999997999999949, 20283007846, 5, 6, -1, 188889999998356876364398
18888999999999997999999949, 3967138802846, 5, 6, -1, 1888899999678661556969398
18888999999999997999999949, 1407826118846, 5, 6, -1, 1888899999885965884373398
18888999999999997999999949, 384879054846, 5, 6, -1, 1888899999968824596557398

```

```
Warning, computation interrupted
```

```
> j;
42830 (32)
```

```
> a := 1888899999999999799999949;
ktop := iquo(a, 81);
r2 := rand(4..19);
d := 5;
for j from 1 to 1000000 do
  r3 := rand(104..4·10r2());
  k := r3()·1000 + 46;
  # k is the number of nines at the end of N1. log-randomized
  if k < ktop then
  # d is the digit that precedes the k digits of nine at the end of N1.
  q := a + 2·d + 1 - k·81;
  if happy(q) > 0 and happy(q + 12) > 0 and happy(q + 22) > 0 and happy(q + 32) > 0
    and happy(q + 42) > 0 and happy(q + 52) > 0 and happy(q + 62) > 0 then print(a,
      k, d, happy(q + 72), happy(q + 82), a - d2 - k·81) end if;
  end if;
end do;
a := 1888899999999999799999949
ktop := 23319753086419750617283
```

```
r2 := proc( )
```

```
proc( ) option builtin = RandNumberInterface; end proc(6, 16, 4) + 4
```

```
end proc
```

```
d := 5
```

```
1888899999999999799999949, 251136599046, 5, 6, -1, 188889999979657735477198
1888899999999999799999949, 14866917792046, 5, 6, -1, 1888899998795779458844198
1888899999999999799999949, 25943470821046, 5, 6, -1, 1888899997898578663495198
1888899999999999799999949, 2645693871046, 5, 6, -1, 188889999785698596445198
1888899999999999799999949, 25943473489046, 5, 6, -1, 1888899997898578447387198
1888899999999999799999949, 28817968046, 5, 6, -1, 18888999997665544588198
```

```
Warning, computation interrupted
```

```
> j;
42392 (33)
```

```
> a := 1888899999999999799999949;
ktop := iquo(a, 81);
r2 := rand(4..20);
d := 5;
for j from 1 to 1000000 do
  r3 := rand(104..4·10r2());
  k := r3()·100 + 46;
  # k is the number of nines at the end of N1. log-randomized
  if k < ktop then
  # d is the digit that precedes the k digits of nine at the end of N1.
  q := a + 2·d + 1 - k·81;
  if happy(q) > 0 and happy(q + 12) > 0 and happy(q + 22) > 0 and happy(q + 32) > 0
    and happy(q + 42) > 0 and happy(q + 52) > 0 and happy(q + 62) > 0 then print(a,
```

```
    k, d, happy(q + 72), happy(q + 82), a - d2 - k·81) end if;  
end if;  
end do:
```

```
    a := 18888999999999997999999949  
    ktop := 23319753086419750617283
```

```
r2 := proc( )
```

```
    proc( ) option builtin = RandNumberInterface; end proc(6, 17, 5) + 4
```

```
end proc
```

```
    d := 5
```

```
18888999999999997999999949, 3915321375507446, 5, 6, -1, 1888899682858968383896798  
18888999999999997999999949, 2782163646, 5, 6, -1, 188889999999774444744598  
18888999999999997999999949, 12409949695746, 5, 6, -1, 1888899998994793874644498  
18888999999999997999999949, 247458211279046, 5, 6, -1, 1888899979955884686397198  
18888999999999997999999949, 202733973754446, 5, 6, -1, 1888899983578547925889798  
18888999999999997999999949, 52027696646, 5, 6, -1, 188889999995785556571598  
18888999999999997999999949, 69642637446, 5, 6, -1, 188889999994358746366798  
18888999999999997999999949, 407768176677854346, 5, 6, -1, 1888866970777688893797898  
18888999999999997999999949, 28078411546, 5, 6, -1, 188889999997725448664698  
18888999999999997999999949, 38916510646, 5, 6, -1, 188889999996847562637598  
18888999999999997999999949, 227558557446, 5, 6, -1, 1888899999981567556846798  
18888999999999997999999949, 27409187988546, 5, 6, -1, 1888899997779855572927698  
18888999999999997999999949, 31112374252046, 5, 6, -1, 1888899997479897485584198  
18888999999999997999999949, 2882272475446, 5, 6, -1, 1888899999766535729488798  
18888999999999997999999949, 19150929946, 5, 6, -1, 188889999998448574674298  
18888999999999997999999949, 15363266846, 5, 6, -1, 188889999998755375385398  
18888999999999997999999949, 30646973348446, 5, 6, -1, 1888899997517594958775798  
18888999999999997999999949, 291927456846, 5, 6, -1, 188889999976353675995398  
18888999999999997999999949, 1896434039646, 5, 6, -1, 1888899999846388642788598  
18888999999999997999999949, 39843980546, 5, 6, -1, 188889999996772437575698  
18888999999999997999999949, 168566954946, 5, 6, -1, 1888899999986345876649298  
18888999999999997999999949, 262260747846, 5, 6, -1, 188889999978756679424398  
18888999999999997999999949, 4499290646, 5, 6, -1, 188889999999635357457598  
18888999999999997999999949, 462368334746, 5, 6, -1, 1888899999962547964885498  
18888999999999997999999949, 55308941029646, 5, 6, -1, 1888899995519975576598598  
18888999999999997999999949, 1743851915746, 5, 6, -1, 1888899999858747794824498  
18888999999999997999999949, 17840056346, 5, 6, -1, 188889999998554755435898  
18888999999999997999999949, 162053395746, 5, 6, -1, 1888899999986873474944498  
18888999999999997999999949, 3090628079446, 5, 6, -1, 1888899999749658925564798  
18888999999999997999999949, 169286880546, 5, 6, -1, 1888899999986287562675698  
18888999999999997999999949, 263138431292346, 5, 6, -1, 1888899978685786865319898  
18888999999999997999999949, 2001771754646, 5, 6, -1, 1888899999837856287873598  
18888999999999997999999949, 14718949791746, 5, 6, -1, 1888899998807764866868498  
18888999999999997999999949, 12895244975946, 5, 6, -1, 1888899998955484956948298
```



```

1888899999999999799999949, 261926507907446, 5, 6, -1, 1888899978783952659496798
1888899999999999799999949, 2720803000546, 5, 6, -1, 188889999779614756955698
1888899999999999799999949, 215063486946, 5, 6, -1, 188889999982579657557298
1888899999999999799999949, 18684495746, 5, 6, -1, 188889999998486355844498
1888899999999999799999949, 257069664448446, 5, 6, -1, 1888899979177356979675798
1888899999999999799999949, 26595588446, 5, 6, -1, 188889999997845557335798
1888899999999999799999949, 29718803492606446, 5, 6, -1, 1888897592776916898877798
1888899999999999799999949, 4460932646, 5, 6, -1, 188889999999638464455598

```

Warning, computation interrupted

```

> a := 1888899999999999799999949;
  ktop := iquo(a, 81);
  r2 := rand(14..19);
  d := 5;
  for j from 1 to 1000000 do
    r3 := rand(1014..3·10r2());
    k := r3()·1000 + 346;
    # k is the number of nines at the end of N1. log-randomized
    if k < ktop then
      # d is the digit that precedes the k digits of nine at the end of N1.
      q := a + 2·d + 1 - k·81;
      if happy(q) > 0 and happy(q + 12) > 0 and happy(q + 22) > 0 and happy(q + 32) > 0
        and happy(q + 42) > 0 and happy(q + 52) > 0 and happy(q + 62) > 0 then print(a,
          k, d, happy(q + 72), happy(q + 82), a - d2 - k·81) end if;
      end if;
    end do;
    a := 1888899999999999799999949
    ktop := 23319753086419750617283

```

```

r2 := proc( )
  proc( ) option builtin = RandNumberInterface; end proc(6, 6, 3) + 14
end proc

```

```

          d := 5
1888899999999999799999949, 1604938835097852346, 5, 6, -1,
1888769999954356873959898
1888899999999999799999949, 127410129641768346, 5, 6, -1, 1888889679779498816763898
1888899999999999799999949, 148766114950003346, 5, 6, -1, 1888887949944688849728898

```

Warning, computation interrupted

```

> j;
          136761

```

(34)

```

> a := 1888899999999999799999949;
  ktop := iquo(a, 81);
  r2 := rand(15..20);
  d := 5;
  for j from 1 to 10000000 do
    r3 := rand(1015..4·10r2());
    k := r3()·100 + 46;

```


1888899999999999799999949, 2940914987653148646, 5, 6, -1,
 1888661785885999894959598
 1888899999999999799999949, 151013581751453346, 5, 6, -1, 1888887767899877932278898
 1888899999999999799999949, 284075063096980546, 5, 6, -1, 1888876989919888944575698
 1888899999999999799999949, 188049531113864246, 5, 6, -1, 1888884767987979576995998
 1888899999999999799999949, 1358128544568217446, 5, 6, -1,
 1888789991587889774386798
 1888899999999999799999949, 126185203220495846, 5, 6, -1, 1888889778998538939836398
 1888899999999999799999949, 271852235073093546, 5, 6, -1, 1888877979968958879422698
 1888899999999999799999949, 322641976065555746, 5, 6, -1, 1888873865999938489984498
 1888899999999999799999949, 140938323950742446, 5, 6, -1, 1888888583995759789861798
 1888899999999999799999949, 297607535822717446, 5, 6, -1, 1888875893789598159886798
 1888899999999999799999949, 362138271617301546, 5, 6, -1, 1888870666799998798574698
 1888899999999999799999949, 247063729751091546, 5, 6, -1, 1888879987837889961584698
 1888899999999999799999949, 293950647590260746, 5, 6, -1, 1888876189997544988879498
 1888899999999999799999949, 188913965712716546, 5, 6, -1, 1888884697968777069959698
 1888899999999999799999949, 250026177195272346, 5, 6, -1, 1888879747879646982939898
 1888899999999999799999949, 186449662972902546, 5, 6, -1, 1888884897577298994893698
 1888899999999999799999949, 158643247557160746, 5, 6, -1, 1888887149896947669979498
 1888899999999999799999949, 26053088890894024746, 5, 6, -1,
 1886789699799837383995498
 1888899999999999799999949, 2596238433454605346, 5, 6, -1, (35)
 1888689704686889976966898

$\gg j;$ 10000001 (36)

The biggest value of k from these random searches is
 1604938835097852346

13875567903719086432546

A search with a separate worksheet yielded a large value of

22507571851987924817546

which is relatively close to the highest possible k of
 23319753086419750617283

We verify that this biggest value of k (so far) does work to give 8 before the carry and 8 after the carry.

$N = 1.(812181234431825799737 \text{ nines}).5.(22507571851987924817546 \text{ nines}).2$
 $\gg \text{evalf}(\log_{10}(21362101260617446049546)); \text{evalf}(\log_{10}(23319753086419750617283));$
22.32964397 (37)

$\gg \text{evalf}(\log_{10}(22507571851987924817546)); \text{evalf}(\log_{10}(23319753086419750617283));$
22.35232864 (38)
22.36772395

```
> 23319753086419750617283 - 22507571851987924817546  
812181234431825799737 (39)
```

```
> minimalbigN(188889999999999799999949 - 52);  
[199999, 23319753086419750617278] (40)
```

```
> 23319753086419750617278 + 5; % - 22507571851987924817546;  
23319753086419750617283  
812181234431825799737 (41)
```

```
> a1 := onestep(1) + 812181234431825799737·81 + 52 + 22507571851987924817546·81;  
for d from 1 to 9 do print(d, happy(a1 + d2)) end do;  
a1 := 188889999999999799999949  
1, -1  
2, 4  
3, 5  
4, 6  
5, 5  
6, 5  
7, 8  
8, 5  
9, 4 (42)
```

```
> a1 := onestep(1) + 812181234431825799737·81 + 62;  
for d from 0 to 8 do print(d, happy(a1 + d2)) end do;  
a1 := 65786679988977889778734  
0, 5  
1, 8  
2, 6  
3, 5  
4, 5  
5, 5  
6, 6  
7, 6  
8, -1 (43)
```