（ＪＡＰＬＡ 2008/01/26）

「幸運数(fortune number)：改定版」について

帝京平成大学　鈴木義一郎

,.″: 13 1 3	,.&.″: 13 1 3	*:,.&″: 13 domain error *:,.&″:13	*:″.,.&″: 13 1 9
*:,.&.″: 13 1 9	+/([:*:,.&.″:)13 10	(next=:[:+/[:*:,.&.″:)13 10	
「″:」は文字化するという演算子、逆演算の「″.」は数値化する演算子である。「next」という関数は、1桁の数値に分解した各数値の平方和を算出し、「次の数」を出力する。 「,.&.″:」のように「&.」という接続詞で接続すると、逆演算が付加され「″.,.&″:」			
next^:1 2(13) 10 1	何回かのこのような“操作”で「１」に推移したとき、「13」を「fortune number」と名づける。		
next^:(1+i.8)37 58 89 145 42 20 4 16 37	最初の数「37」に戻ってしまったので、この後は同じ数字を循環するだけだから１に推移することはないので、「unfortune(number)」と名づける。		

次に右引数で与えた数に対して、１に推移する(fortune)か、循環列に入った(unfortune)ところで「stop」する片側関数を、次のように定義する。

```
   reg_no=:3 :0
k=.#s=.r=.(next=.[:+/[:*:,.&.″:)y
while. k=1
  do. k=.s-:~.s=.s,r=.next r
end.
s
)
```

```
   reg_no 2
4 16 37 58 89 145 42 20 4
   reg_no 3
9 81 65 61 37 58 89 145 42 20 4 16 37
   reg_no 4
16 37 58 89 145 42 20 4 16
   reg_no 7
49 97 130 10 1 1
   reg_no 10
1 1
   reg_no 11
2 4 16 37 58 89 145 42 20 4
   reg_no 12
5 25 29 85 89 145 42 20 4 16 37 58 89
   reg_no 13
```

	10 1 1

更に、「100」までこの関数を適用した結果から、循環列に入るケースは

c=:4 16 61 20 24 42 37 73 58 85 89 98 145 154 415 451 514 541

といった数値のいずれかに推移したときに循環列に入ることが確かめられる。

次に、右引数で与えた数字以下の fortune number を出力する片側関数を以下のように定義する。

fortune0=:3 :0 f=.r=.1 while. r<y do. f=.f,(1={:reg_no r)#r=.r+1 end.)	fortune=:3 :0 M. f=.r=.1 while. r<y do. f=.f,(1={:reg_no r)#r=.r+1 end.)

6!:2'fortune0 15' 0.00913105 6!:2'r15=:fortune 15' 0.00015421	6!:2'fortune0 100' 0.0507316 6!:2'r100=:fortune 100' 0.000212597	

r15 1 7 10 13	r100 1 7 10 13 19 23 28 31 32 44 49 68 70 79 82 86 91 94 97 100

以下では、fortune number の個数だけを調べてみる。

6!:2'# fortune0 1000' 0.464007 6!:2'n1=:# fortune 1000' 0.468065	6!:2'# fortune0 10000' 4.71541 6!:2'n2=:# fortune 10000' 4.51666
6!:2'# fortune0 100000' 46.5121 6!:2'n3=:# fortune 100000' 46.476	6!:2'# fortune0 1000000' 530.787 6!:2'n4=:# n=:fortune 1000000' 536.751

n1,n2,n3,n4 143 1442 14377 143071	以上の結果から、fortune number の個数の割合は、１４％強含まれていることが分かる。なお、「M.」の利用が計算時間の短縮にはほとんど寄与しないことが検証された。また、バージョン「J504」での最後の結果の演算時間は 609.72 秒であった。

さらに、fortune number の個数が 10 等分した範囲にどの程度一様に分布しているかについて調べてみると、「ｎ」には 1000000 以下の fortune number が入力されているから

```
   +/"1=<.(143{.n)%100
19 13 12 22 10 5 19 11 13 18 1
   +/"1=<.(1442{.n)%1000
142 156 144 162 130 150 129 149 149 130 1
   +/"1=<.(14377{.n)%10000
1441 1597 1601 1448 1537 1464 1237 1434 1338 1279 1
   +/"1=<.n%100000
14376 14861 15003 14341 14205 14386 13841 14313 14172 13572 1
```

次の関数は、fortune number か unfortune(number)かを判定し、さらに判定に至るまでに要したステップ数と循環列にはいったときの数を出力するものである。

```
   judge=:3 :0

s=.r=.(next=.[:+/[:*:,.&."):y
c=.4 16 61 20 24 42 37 73 58 85
c=.c,89 98 145 154 415 451 514 541
while.-.r e.1,c
  do. s=.s,r=.next r
end.
if.r=1 do.(#s);'fortune'
       else.(#s);r;'unfortune'
end.
)
```

```
   judge 2

1 | 4 | unfortune     Judge 6
8 | 85 | unfortune    Judge 7
5 | fortune          judge 13
2 | fortune          judge 77
1 | 98 | unfortune
```

```
   judge 9999

3 | 85 | unfortune
```

```
   Judge 99999

7 | 85 | unfortune
```

```
   judge 899999

7 | fortune
```

以上の適用結果をみると、数字が大きくなっても fortune number か unfortune(number)かの判定に要するステップ数は存外多くはならないことが予想される。

一般に、ｐ進数の場合についても同じ問題を考えてみよう。

10進数の数値をｐ進数に変換する関数を次のように定義する。

```
p_10=:13 :'((>.x^.1+y)$x)#:y'
```

```
regp_no=:4 :0

p_10=.(([:>.[^.1:+])$[)#:]
(t=.q=.+/*:r);s=.<r=.y [ d=.1
while. d=1 do.
  (t=.t,q=.+/*:r);s=.s,<r=.x  p_10 q
  (t;s) [ d=.t-:~.t
end.
)
```

```
   5 regp_no 1 1
```

2 4 16 10 4	1 1	2	4	3 1	2 0	8

```
regp_no 1 1
```

2 4 16 4	1 1	2	4	2 0	8

```
regp_no 1 2
```

5 25 10 5	1 2	5	3 1	1 2	8

```
regp_no 1 3
```

10 5 25 10	1 3	1 2	5	3 1	$[1\,3]_8$ は

「１」に収束しているので fortune number である。

```
   8 regp_no 1 4
```

17 5 25 10 5	1 4	2 1	5	3 1	1 2	8

```
regp_no 1 5
```

26 13 26	1 5	3 2	1 5	8

```
regp_no 1 6
```

37 41 26 13 26	1 6	4 5	5 1	3 2	1 5	8

```
regp_no 1 7
```

50 40 25 10 5 25	1 7	6 2	5 0	3 2	1 2	5

```
   8 regp_no 2 0
```

4 16 4	2 0	4	2 0	8

```
regp_no 2 1
```

5 25 10 5	2 1	5	3 1	1 2	

```
8 regp_no 2 2
```

8 1 1	2 2	1 0	1	$[2\,2]_8$ は

「１」に収束しているので fortune number である。

```
   8 regp_no 2 3
```

13 26 13	2 3	1 5	3 2	

```
8 regp_no 2 4
```

20 20	2 4	2 4	

```
   8 regp_no 2 5
```

29 34 20 20	2 5	3 5	4 2	2 4	8 regp_no 2

```
6
```

40 25 10 5 25	2 6	5 0	3 1	1 2	5	

```
   8 regp_no 7 1
```

50 40 25 10 5 25	7 1	6 2	5 0	3 1	1 2	5

8 regp_no 7 2

53 61 74 6 36 32 16 4 16	7 2	6 5	7 5	1 1 2	6	4 4	4 0	2 0	4

8

regp_no 7 3

58 53 61 74 6 36 32 16 4 16	7 3	7 2	6 5	7 5	1 1 2	6	4 4	4 0	2 0	4

8 regp_no 7 4

65 2 4 16 4	7 4	1 0 1	2	4	2 0

8 regp_no 7 5

74 6 36 32 16 4 16	7 5	1 1 2	6	4 4	4 0	2 0	4

8 regp_no 7 6

85 30 45 50 40 25 10 5 25	7 6	1 2 5	3 6	5 5	6 2	5 0	3 1	1 2	5

8

regp_no 7 7

98 21 29 34 20 20	7 7	1 4 2	2 5	3 5	4 2	2 4

最後に、右引数に10進法で与えた数以下のｐ進数の fortune number を出力する両側関数を次のように定義する。(以下では、バージョン「J504」での演算結果である。)

```
    fortunep=:4 :0

 p_10=.(([:>.[^.1:+])$[)#:]
 r=.1+#f=.''
 while. r<y.
   do. t=.x.regp_no x.p_10 r=.r+1
       f=.f,(1=+/*:,.&.":{:>{.t)#1{t
 end.
)
```

8 regp_no 1 1 3

11 10 5 25 10	1 1 3	1 3	1 2	5	3 1

8 regp_no 1 1 4

18 8 1 1	1 1 4	2 2	1 0	1

8

regp_no 1 1 5

27 18 8 1 1	1 1 5	3 3	2 2	1 0	1