International Journal of Mathematical Archive-7(6), 2016, 126-134
Available online through www.ijma.info ISSN 2229 – 5046

KAPREKAR NUMBERS AND ITS ANALOG EQUATIONS

KAILASH M. PATIL*

Assistant Professor, Mathematics Department,
Dharmsinh Desai University, Nadiad, Gujarat, India.

NIKHIL P. SHAH
Assistant Professor, MCA Department,
Dharmsinh Desai University, Nadiad, Gujarat, India.

(Received On: 13-06-16; Revised & Accepted On: 28-06-16)

ABSTRACT

The present article discusses the contribution of D.R.Kaprekarin a recreational number theory, particularly Kaprekar Constant and Kaprekar Number. The analog equations of Kaprekar number

1. $K^3 = P_0 + P_1(10) + P_2(10^2) + P_3(10^3) + ... + P_n(10^n)$
   $\mathrm{K} = P_0 + P_1 + P_2 + P_3 + ... + P_n$
2. $K^3 = P(10^n) - Q$; $0 \le Q \le P$
   $\mathrm{K} = P - Q$
3. $\mathrm{K}^3 = P(10^n) + Q$
   $\mathrm{K} = \mathrm{P} + \mathrm{Q}$; *where* P *and* Q *are positive integers,* $0 < \mathrm{Q} < \mathrm{P}$ *and* $\mathrm{n} \ge 1$.

are discussed. C Programs are developed for the same and for Kaprekar Number the program is dynamic is enough to suit any base.

***Keywords**: Kaprekar Constant, Kaprekar Number, Kaprekar Cycle, Reverse subtraction process.*

1. INTRODUCTION

D.R.Kaprekar was born on January 17, 1905 in Maharashtra. For whatever reason he is not so well known in Mathematical world but his contribution in recreation mathematics cannot be ignored. He discovered Kaprekar Constant, Kaprekar number, Self-numbers, Harshad numbers, Demlo numbers and many more such numbers. In this paper C program for Kaprekar Constant, Kaprekar number and its analog equations are discussed which can be considered as a tribute to D.R.Kaprekar. The C program developed in this paper is dynamic enough to get Kaprekar number for any base 10, 2(Binary), 8(Octal) and so on.

2. KAPREKAR CONSTANT AND KAPREKAR PROCESS

Reverse Subtraction Process

If we consider any 4-digit number where all the digits should not be alike, then we can generate two different numbers by arranging them into ascending and descending order. Subtraction of smaller number from larger number will generate another four digit number. Occasionally, a three digitsnumber will be generated then put zero on the extreme left side of the number. This process is called as reverse subtraction process.

Now the question arises that if we keep on repeating the above said reverse subtraction process, Can we have an infinite sequence of different numbers? Or a sequence of finite numbers.

Corresponding Author: Kailash M. Patil*
Assistant Professor, Mathematics Department,
Dharmsinh Desai University, Nadiad, Gujarat, India.

Let us consider the following example:

Number = 3947
Step 1: Number = 3947
Ascending of Number: 3479
Descending of Number: 9743
Reverse Subtraction Process: 6264

Step 2: Number = 6264
Ascending of Number: 2466
Descending of Number: 6642
Reverse Subtraction Process: 4176

Step 3: Number = 4176
Ascending of Number: 1467
Descending of Number: 7641
Reverse Subtraction Process: 6174

Step 4: Number = **6174**
Ascending of Number: 1467
Descending of Number: 7641
Reverse Subtraction Process: **6174**

Now, there is no use of repeating the process as step3& step4 are same. Therefore starting from 3947, a finite sequence of numbers {3947, 6264, 4176and 6174} will be generated with the help of Reverse Subtraction Process. The interesting fact is that if we start with any four digit number (not all the digits are same) then within eight or less steps the sequence converges to 6174, which is known as Kaprekar Constant. The same is developed here in C program.

Program for Kaprekar Process [Kaprekar Constant]

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#include <values.h>
voidAscendingOrder(int ascending[],int n)
{
	inti,k,y;
	for(k=1;k<n;k++)
	{
		y=ascending[k];
		for(i=k-1;i>=0 && y<ascending[i]; i--)
			ascending[i+1]=ascending[i];
		ascending[i+1]=y;
	}
}
voidDescendingOrder(int descending[],int n)
{
	inti,k,y;
	for(k=1;k<n;k++)
	{
		y=descending[k];
		for(i=k-1;i>=0 && y>descending[i]; i--)
			descending[i+1]=descending[i];
		descending[i+1]=y;
	}
}
//Function is used to conver number into array
int * numberToArray(int number)
{
	inti = 0,*arr=NULL;
	arr=(int *)malloc(4);
	if(number<1000 && number>99)
	{
```

```
arr[i]=0;
                i++;
        }
        if(number<100 && number>9)
        {
                arr[i]=0;
                arr[++i]=0;
                i++;
        }
        if (number>1 && number<10)
        {
                arr[i]=0;
                arr[++i]=0;
          //    i++;
        }
        while (number > 0)
        {
                arr[i] = number % 10;
                number /= 10;
                i++;
        }
        returnarr;
}
void main(void)
{
        inti,sum=0,cnt=0,temp=MAXINT;
        int *arr,*arr1,ans1[10],n,ans,n1=0,n2=0;
        clrscr();
        printf("Enter a Number:");
        scanf("%d",&n);
        arr = numberToArray(n);
con:
        arr1 = arr;
        printf("\n");
        DescendingOrder(arr,4);
        for(i=0;i<4;i++)
        {
                printf(" %d ",arr[i]);
                n1 = 10 * n1 + arr[i];
        }
        printf("\n");
        AscendingOrder(arr1,4);
        for(i=0;i<4;i++)
        {
                printf(" %d ",arr1[i]);
                n2 = 10 * n2 + arr1[i];
        }
        printf("\n");
        ans = n1-n2;
        printf("\n Answer is %d",ans);
        getch();
        if (ans<1000)
        {       }
        if (ans==temp)
                gotoexi;
        else
        {
                cnt++;
                temp=ans;
                *arr=NULL;
                n1=n2=0;
                arr = numberToArray(ans);
                goto con;
        }
```

```
exi:
printf("\n No. of Iterations %d",cnt);
            getch();
}
/* OUTPUT
Enter a Number: 1234
4  3  2  1
1  2  3  4
Answer is 3087
8  7  3  0
0  3  7  8
Answer is 8352
8  5  3  2
2  3  5  8
Answer is 6174
7  6  4  1
1  4  6  7
Answer is 6174
 No. of Iterations 3
*
/
```

3. KAPREKAR CYCLE

Note that the above Kaprekar process is not applicable for single digit number but it is applicable for more than one digit numbers. For two digit number the Kaprekar Process is:

Number: 89

1. Ascending Number : 98
   Descending Number : 89
   Subtraction : 09

2. Ascending Number : 09
   Descending Number : 90
   Subtraction : 81

3. Ascending Number : 18
   Descending Number : 81
   Subtraction : 63

4. Ascending Number : 36
   Descending Number : 63
   Subtraction : 27

5. Ascending Number : 27
   Descending Number : 72
   Subtraction : 45

6. Ascending Number : 45
   Descending Number : 54
   Subtraction : 09

Note that step 2 & step 6 are same. This process generates Kaprekar Cycle 89 → 09 →81 →63 → 27 → 45 → 09. Similarly, one can have a Kaprekar Cycle for other digits also but for four digits we get Kaprekar cycle of length 1. Amazing fact is Kaprekar constant or number belonging to Kaprekar Cycle are divisible by 9, see [5].

4. KAPREKAR NUMBER

Number K is said to be Kaprekar number if K^2 is divided into two parts, left and right, such that the sum of the two parts is equal to K. Mathematically, K is a Kaprekar number if

$$K^2 = P * 10^n + Q \left(n >= 1, P >= 0. 0 < Q < 10^n \right)$$

and

$$K = P + Q$$

Therefore,

$$(P + Q)^2 = P(10^n) + Q$$

$$\Rightarrow P^2 + 2PQ + Q^2 - (10^n)P + Q = 0$$

$$\Rightarrow P^2 + P(2Q - 10^n) + Q^2 + Q = 0$$

$$\Rightarrow P = \frac{10^n - 2Q \pm \sqrt{10^{2n} - 4Q(10^n) + 4Q}}{2}$$

Also,

$$10^{2n} - 4Q(10^n - 1) \geq 0$$

$$\Rightarrow 10^{2n} \geq 4Q\ (10^n - 1)$$

$$\Rightarrow 0 \leq Q \leq \frac{10^{2n}}{4(10^n - 1)},$$ give the upper bound for Q.

For the fixed value of n, the C program developed is based on the following

1. $10^{2n} - 4Q\ (10^n - 1)$ is a perfect square
2. The upper bound of Q is $\frac{10^{2n}}{4(10^n - 1)}$.

The size of data type in C depends on machine so the existing program execution depends on the machine. Some examples of Kaprekar numbers are

9^2= 81 and 9= 8 + 1

5050^2 = 25502500 and 5050 = 2550+2500

703^2 = 494209 and 703 = 494 + 209

Observed that for any Kaprekar number there exists another Kaprekar number such that $K_1 + K_2 = 10^n$ where n is a positive integer.

<u>Program for Kaprekar Numbers</u>

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<process.h>
void main()
{
        double n,q,div,div1,squareroot,p,p1,perfectsqrt[100],qq[100];
        inti=0,j=0,k,k1,cnt=0,iteration,psqrt;
        clrscr();
        printf("Enter the value of n:");
        scanf("%lf",&n);
        div = pow(10,2*n);
        div1 = 4*(pow(10,n)-1);
        iteration = div / div1;
iteration ++;
        for(q=0;q<iteration;q++)
        {
                squareroot = sqrt(div-(q*div1));
                psqrt = squareroot;
                if(squareroot == psqrt)
                {
                qq[i++] = q;
                perfectsqrt[j++] = squareroot;
                        cnt++;
                }
```



```
        }
        for (i=0;i<cnt;i++)
        {
                p=(pow(10,n)-(2*qq[i])+perfectsqrt[i])/2;
                p1=(pow(10,n)-(2*qq[i])-perfectsqrt[i])/2;
                k = p + qq[i];
         k1 = p1 + qq[i];
                printf("\nPerfect Squares (P + Q) is %d = %lf + %0.2lf",k,p,qq[i]);
                printf("\nPerfect Squares (P + Q) is %d = %lf + %0.2lf",k1,p1,qq[i]);
        }
        getche();
}
/*OUTPUT
Enter the value of n: 3
Perfect Squares (P + Q) is 1000 = 1000.000000 + 0.00
Perfect Squares (P + Q) is 0 = 0.000000 + 0.00
Perfect Squares (P + Q) is 999 = 998.000000 + 1.00
Perfect Squares (P + Q) is 1 = 0.000000 + 1.00
Perfect Squares (P + Q) is 703 = 494.000000 + 209.00
Perfect Squares (P + Q) is 297 = 88.000000 + 209.00
*/
```

The above program can be used for any other base by just replacing 10 by the required base number. The following are the sample outputs for the binary base 2.

Sample Output for binary base

```
Enter the value of n: 3
Perfect Squares (P + Q) is 8 = 8.000000 + 0.00
Perfect Squares (P + Q) is 0 = 0.000000 + 0.00
Perfect Squares (P + Q) is 7 = 6.000000 + 1.00
Perfect Squares (P + Q) is 1 = 0.000000 + 1.00

Enter the value of n: 4
Perfect Squares (P + Q) is 16 = 16.000000 + 0.00
Perfect Squares (P + Q) is 0 = 0.000000 + 0.00
Perfect Squares (P + Q) is 15 = 14.000000 + 1.00
Perfect Squares (P + Q) is 1 = 0.000000 + 1.00
Perfect Squares (P + Q) is 10 = 6.000000 + 4.00
Perfect Squares (P + Q) is 6 = 2.000000 + 4.00

Enter the value of n: 10
Perfect Squares (P + Q) is 1024 = 1024.000000 + 0.00
Perfect Squares (P + Q) is 0 = 0.000000 + 0.00
Perfect Squares (P + Q) is 1023 = 1022.000000 + 1.00
Perfect Squares (P + Q) is 1 = 0.000000 + 1.00
Perfect Squares (P + Q) is 837 = 684.000000 + 153.00
Perfect Squares (P + Q) is 187 = 34.000000 + 153.00
Perfect Squares (P + Q) is 682 = 454.000000 + 228.00
Perfect Squares (P + Q) is 342 = 114.000000 + 228.00
Perfect Squares (P + Q) is 528 = 272.000000 + 256.00
Perfect Squares (P + Q) is 496 = 240.000000 + 256.00
*/
```

Some of the examples for binary base are

1. $7 = (6)_2 + 1_2$ and $7^2 = 49 = (6)_2 (2^3) + 1_2$ that is
   $7 = 110 + 001$ and $7^2 = 49 = 110001$
2. $837 = 684_2 + 153_2$ and $837^2 = 700569 = (684)_2(2^{10}) + 153$
   $837 = 1010101100 + 1001\ 1001$ and $837^2 = 101010110010011001$

Observed that for any Kaprekar number with base 2 there exists another Kaprekar number with base 2 such that $K_1 + K_2 = 2^n$, where n is a positive integer.

5. ANALOG EQUATIONS OF KAPREKAR NUMBER

5.1 $K^3 = P_0 + P_1(10) + P_2(10^2) + P_3(10^3) + \ldots + P_n(10^n)$

$K = P_0 + P_1 + P_2 + P_3 + \ldots + P_n$

<u>Program for 5.1</u>

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
long double x[150];
longinti, sum,n,a[150];
        clrscr();
printf("Enter the value of n");
scanf("%ld",&n);
for(i=0;i<=n;i++)
    {
        a[i]=pow(i,3);
while(a[i]>0)
         {
x[i] = fmod(a[i],10);
sum = sum + x[i];
a[i] = a[i] / 10;
         }
if(sum == i)
        printf("\n i = %ld sum = %ld",i,sum);
sum=0;
    }
        getche();
}
/*OUTPUT
Enter the value of n100
i = 1 sum = 1
i = 8 sum = 8
i = 17 sum = 17
i = 18 sum = 18
i = 26 sum = 26
i = 27 sum = 27
*/
```

Observed that

8^3= 512 and 8 = 5 + 1 + 2

26^3 = 17576 and 26 = 1 +7 +5 + 7 + 6

Surprisingly, such numbers are very few and above output gives all such numbers.

5.2 $K^3 = P(10^n) - Q\ ;\ 0 \le Q \le P$

$K = P - Q$

Therefore,

$$K^3 - K = P(10^n - 1)$$

$$\Rightarrow K(K-1)(K+1) = P(10^n - 1)$$

Thus, all the Kaprekar numbers will satisfy the above mentioned equation. Some of the examples are:

55^3= 166375 = 1680(10^2) – 1625 and 55 = 1680 – 1625

703^3= 347428927 = 347776 (10^3) – 347073 and 703 = 347776 – 347073

Clearly, the density of such numbers may be more than Kaprekar Numbers since there may exists numbers which are not Kaprekar number but may satisfy the above equation.

5.3 $K^3 = P(10^n) + Q$

$K = P + Q$; where, P and Q are positive integers, $0 < Q < P$ and $n \geq 1$.

Therefore, $(P + Q)^3 = P(10^n) + Q$

$$P^3 + Q^3 + 3P^2Q + 3PQ^2 - P(10^n) - Q = 0$$

$$P^3 + 3P^2Q + P[3Q^2 - 10^n] + Q^3 - Q = 0$$

Sum of roots $= -3Q$

Diminishing the roots by $-Q$

$$
\begin{array}{r|llll}
-Q & 1 & 3Q & 3Q^2 - 10^n & Q^3 - Q \\
 & 0 & -Q & -2Q^2 & -Q^3 + Q10^n \\
\hline
 & 1 & 2Q & Q^2 - 10^n & Q10^n - Q \\
 & 0 & -Q & -Q^2 & \\
\hline
 & 1 & Q & -10^n & \\
 & 0 & -Q & & \\
\hline
 & 1 & 0 & &
\end{array}
$$

Therefore, transformed equation is

$$y^3 - 10^n y + Q(10^n - 1) = 0$$

Let $y = u + v$

Therefore, $y^3 - 3uvy - (u^3 + v^3) = 0$

Comparing, $u^3 + v^3 = 10^{3n}/27$ and $u^3 + v^3 = -Q(10^n - 1)$

If u^3 and v^3 are the root of equation $t^2 + Q(10^n - 1) + \frac{10^{3n}}{27} = 0$

$$\Rightarrow t = \frac{-Q(10^n - 1) \pm \sqrt{Q^2(10^n - 1)^2 - 4\frac{10^{3n}}{27}}}{2}$$

Here, $10^{3n}/27$ cannot be an integer, hence, no positive integers P and Q exists for which equation 5.3 is satisfied except Q = 0.

6. CONCLUDING REMARKS

In the Kaprekar process, the most amazing part is the number 6174. But if one thinks logically then for any n digit number if one can able to define the process(Reverse Addition, division and so on.) which generates another n digit number uniquely then process has to terminate after finitely many steps, since we have finite n digit numbers. It means the sequence has to hit a previously generated number. In any case we can have a cycle or a constant. Coincidently, for four digit number the reverse subtraction process gives a cycle of length 1.The analog equations for the Kaprekar numbers are defined and discussed. Many more such equations can be defined in future.

7. REFERENCES

1. D.R.Kaprekar. An interesting property of the number 6174. Scripta Mathematica, 15:244–245, 1955.
2. D.R.Kaprekar. On Kaprekar numbers. Journal of Recreational Mathematics, 13(2):81–82, 1980.
3. D R Kaprekar, The Mathematics of New self-numbers (booklet), published by D R Kaprekar, Devlali, 1962.
4. P. Chaudhury, The Tale of a Neglected Mathematician, his works and beyond., Everyman's Science, Vol XXXIX, Feb – Mar 2005.

5. Tanvir Prince, Kaprekar Constant Revisited, International Journal of Mathematical Archive-4(5), 2013, 52-58
6. World Wide Web. Life of Kaprekar. http://en.wikipedia.org/wiki/Kaprekar, 2000.
7. World Wide Web. The number 6174. http://en.wikipedia.org/wiki/6174, 2000.
8. World Wide Web. Kaprekar series generator. http://kaprekar.sourceforge.net, 2003. Online program to download.
9. World Wide Web. The mysterious 6174 revisited.http://mathpoint.blogspot.com/2006/12/mysterious, 2006

Source of support: Nil, Conflict of interest: None Declared