Editorial [DEEP MATRIX]

Sunday 29/03/2020 – 9:30 P.M to 12:30 P.M

Challenge	Success Rate
1. Code Language	68.42%
2. !_Binary_Clock_!	22.22%
3. !_GREEDY_GRID!	00.00%
4. !_()_FUN_()_!	NA*

*You will be not judged according to success for this question due to wrong test case. Those who have submitted partial correct code have already got points and should consider it as final.

1.Code Language

While communicating with one of your close friends you decided to encrypt your secret message so that in case even if your message reaches invalid person he/she won't be able to perceive it. You and your friend decided to keep a secret key k(positive integer) for communicating with one another. The key is that integer which forwards each of your alphabet by k. eg: if k=2 than b will appear d, z will appear b, y will appear a, a will appear c and so on.

Note : Only capital and small letters gets changed whereas numbers,spaces,etc. don't vary accordingly.

After your negotiation you realized its extremely difficult to perform encryption manually. Hence you decided to program it out.

Program : Your program must scan the key k initially and then input string (plain text) and henceforth display appropriate encrypted message at output (encrypted text).

We ensure input text wont exceed 100 characters and the input string starts with alphabet always.

Input Format

Integer k followed by input string.

Constraints

0

Output Format

Encrypted Output.

Sample Input 0

```
2I love U
```

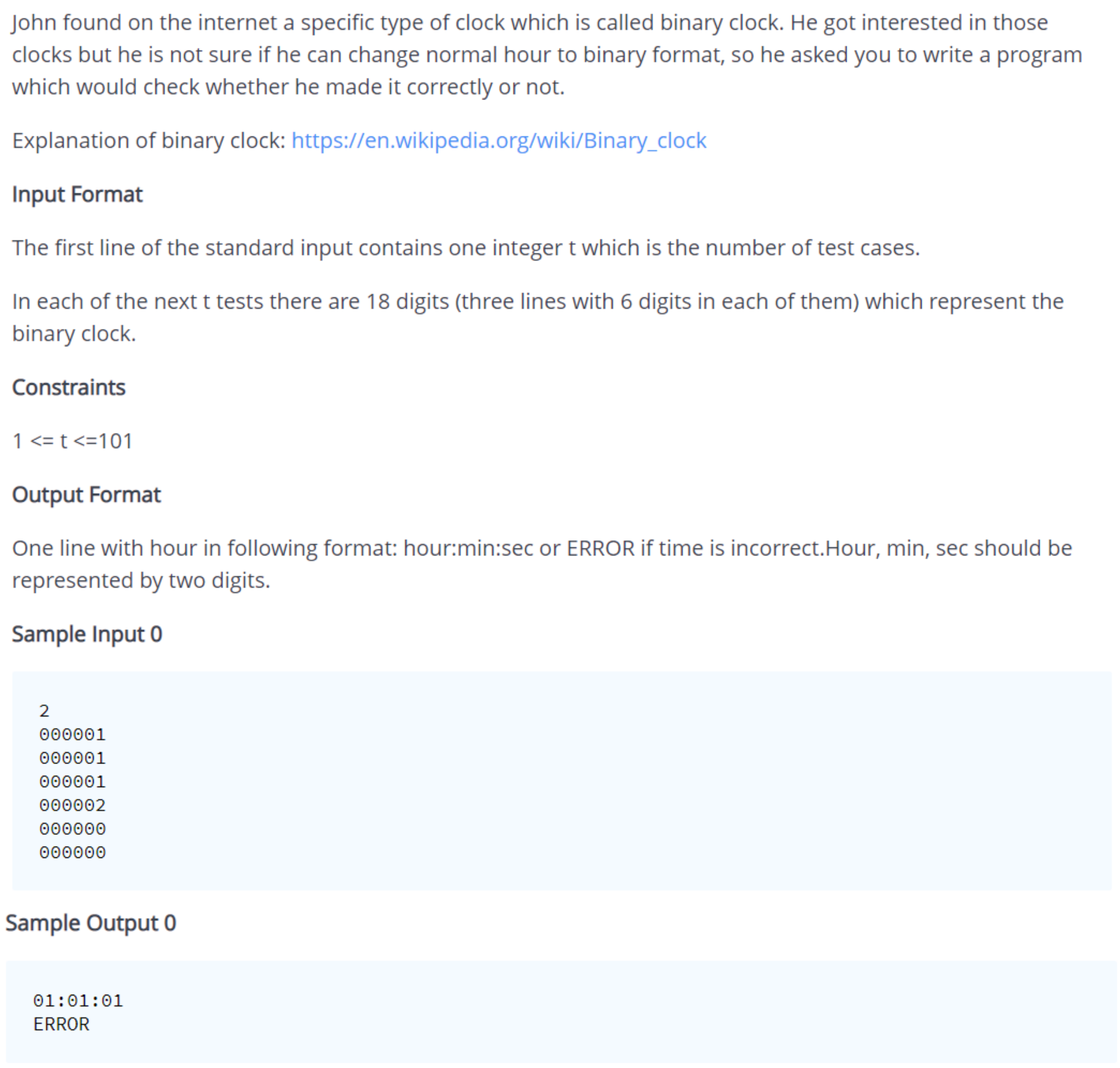
Sample Output 0

```
K nqxg W
```

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main()
{
    char p[500];
    int k,i;
    scanf("%d",&k);
    gets(p);
    while(p[i]!='\0')
    {
         if(p[i]>=65&&p[i]<=90)
         {
            p[i]+=k;
            if(p[i]>90)
p[i]=(p[i]-90+64);
         }
         else if(p[i]>=97&&p[i]<=122)
         {
             p[i]+=k;
             if(p[i]>122)
                 p[i]=(p[i]-122+96);
         }
        i++;
    }
    puts(p);
    return 0;
}
```

2.! Binary Clock !



John found on the internet a specific type of clock which is called binary clock. He got interested in those clocks but he is not sure if he can change normal hour to binary format, so he asked you to write a program which would check whether he made it correctly or not.

Explanation of binary clock: https://en.wikipedia.org/wiki/Binary_clock

Input Format

The first line of the standard input contains one integer t which is the number of test cases.

In each of the next t tests there are 18 digits (three lines with 6 digits in each of them) which represent the binary clock.

Constraints

1 <= t <=101

Output Format

One line with hour in following format: hour:min:sec or ERROR if time is incorrect.Hour, min, sec should be represented by two digits.

Sample Input 0

```
2
000001
000001
000001
000002
000000
000000
```

Sample Output 0

```
01:01:01
ERROR
```

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main() {

    long int tc,i,j,k,p,q,r,dmin,dsec,dhr;
    scanf("%ld",&tc);
    for(i=0;i<tc;i++)
    {
        long int flg=0;
        scanf("%ld %ld %ld",&p,&q,&r);

        dsec=0;dmin=0;dhr=0;
        long int mf=1;
        while(p>0)
        {
            if(p%10>1)
            {   break;
                flg=1;
            }
            dhr=dhr+(p%10)*mf;
            mf*=2;
            p/=10;
        }
        mf=1;
        while(q>0)
        {
            if((q%10>1)||(flg==1))
            {   break;
                flg=1;
            }
            dmin=dmin+(q%10)*mf;
            mf*=2;
            q/=10;
        }
        mf=1;
        while(r>0)
        {
            if((r%10>1)||(flg==1))
            {   break;
                flg=1;
            }
            dsec=dsec+(r%10)*mf;
            mf*=2;
            r/=10;
        }
        if((dsec>=0&&dsec<60)&&(dmin>=0&&dmin<60)&&(dhr>0&&dhr<=24)&&(flg!=1))
        {
            if(dhr<10)
                printf("0%d:",dhr);
            else
                printf("%d:",dhr);
            if(dmin<10)
                printf("0%d:",dmin);
            else
                printf("%d:",dmin);
            if(dsec<10)
                printf("0%d",dsec);
            else
                printf("%d",dsec);
        }
        else
        {
             printf("ERROR");
        }
        printf("\n");
    }
    return 0;
}
```

3.! GREEDY GRID !

You are given n x n grid and a number k. You have to find greatest product of k adjacent numbers in the same direction (up, down, left, right, or diagonally) in the n x n grid.

Input Format

n k
$A_{(0,0)}$ $A_{(0,1)}$ $A_{(0,2)}$ … $A_{(0,n-1)}$
$A_{(1,0)}$ $A_{(1,1)}$ $A_{(1,2)}$ … $A_{(1,n-1)}$
. . . .
. . . .
. . . .
. . . .
$A_{(n-1,0)}$ $A_{(n-1,1)}$ $A_{(n-1,2)}$ … $A_{(n-1,n-1)}$

Constraints

$1 <= k <= n <= 10^4$

Output Format

Greatest product of k adjacent numbers as given in question.

Sample Input 0

```
20 4
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

Sample Output 0

```
70600674
```

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main() {

    long long i,j,n,k,max=-1,m1,m2,m3,m=-1;

    scanf("%lld %lld",&n,&k);

    long long arr[n][n];

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%lld",&arr[i][j]);
        }
    }

    for(i=0;i<n;i++)
    {
        for(j=0;j<n-k+1;j++)
        {
            m1=1;
            for(m=0;m<k;m++)
            {
                m1*=arr[i][j+m];
            }
            max=max<m1?m1:max;
        }
    }

    for(i=0;i<n-k+1;i++)
    {
        for(j=0;j<n-k+1;j++)
        {
            m1=1;
            for(m=0;m<k;m++)
            {
                m1*=arr[i+m][j+m];
            }
            max=max<m1?m1:max;
        }
    }

    for(i=0;i<n-k+1;i++)
    {
        for(j=0;j<0;j++)
        {
            m1=1;
            for(m=0;m<k;m++)
            {
                m1*=arr[i+m][j];
            }
            max=max<m1?m1:max;
        }
    }


    for(i=0;i<n-k+1;i++)
    {
        for(j=k-1;j<n;j++)
        {
            m1=1;
            for(m=0;m<k;m++)
            {
                m1*=arr[i+m][j-m];
            }
            max=max<m1?m1:max;
        }
    }


    printf("%lld\n",max);

    return 0;

}
```

4.! () FUN () !

Let a function be defined as follows:

$f_k(n) = \sum^{n}_{i=0} f_k(\lfloor i/k \rfloor)$ where $f_k(0) = 1$ and $\lfloor x \rfloor$ denotes the floor function.

For each test case t find the value of $f_k(n)$ when n and k is given on each seperate line seperated by a space.

Input Format

t
n k
.
.
.
t times

Constraints

$1 <= t <= 100$
$1 <= n < k^2 <= 10^5$

- **10 points for test cases passing n <= k**
- **10 points for test cases passing $k < n < k^2$**

Output Format

$f_k(n)$
.
.
.
t times

Sample Input 0

```
1
10 5
```

Sample Output 0

```
18
```

Explanation 0

Solve the function for n=10 and k=5 to know.

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main() {

    long long t,n,k,sum;

    scanf("%lld",&t);

    while(t--)
    {
        scanf("%lld %lld",&n,&k);

        if(n==k)
            printf("%lld\n",n+2);
        else if(n<k)
            printf("%lld\n",n+1);
        else{
            sum=0;
            for(long long i=0;i<=n;i++)
                sum+=(i/k+1);
            printf("%lld\n",sum);
        }
    }

    return 0;
}
```

This question was from well known series Project Euler and it has 85% hardness. Due to applied constraints i.e n < k^2 this question became just a straight forward pattern recognizing question as you can see in solution.

We can see that when k=1, there is no solution for this problem, and that is why many submissions went wrong for 3rd test for this question.

```
Input- 3                          Output- 7
      5 5                                 5
      5  4                                2
      7 1
```

This was because test cases were developed using random integer in Python which choose k as random integer 1 due to constraints. Whoever's mistake might be lets forget it and understand concept behind it.