Ex. No. : **Date :**

Suppandi & Areas

Problem Statement:

Suppandi is trying to take part in the local village math quiz. In the first round, he is asked about shapes and areas. Suppandi, is confused, he was never any good at math. And also, he is bad at remembering the names of shapes. Instead, you will be helping him calculate the area of shapes.

- When he says rectangle, he is actually referring to a square.
- When he says square, he is actually referring to a triangle.
- When he says triangle, he is referring to a rectangle
- And when he is confused, he just says something random. At this point, all you can do is say 0.

Help Suppandi by printing the correct answer in an integer.

Input Format

- Name of shape (always in upper case R --> Rectangle, S --> Square, T --> Triangle)
- Length of 1 side
- Length of other side

  Note: In case of triangle, you can consider the sides as height and length of base

Output Format

- Print the area of the shape.

Sample Input 1

T
10
20

Sample Output 1

200

Program:

```
#include<stdio.h>
int main()
{
   int a,b;
   char c;
   scanf("%c%d%d", &c, &a, &b);
   switch(c)
   {
      case 'R':
      printf("%d", a*b);
      break;
      case 'S':
      printf("%0f", (0.5)*a*b);
      break;
      case 'T':
      printf("%d", a*b);
      break;
      default:
      printf("0");
   }
}
```

Ex. No. : **Date :**

Superman's Encounter

Problem Statement:

Superman is planning a journey to his home planet. It is very important for him to know which day he arrives there. They don't follow the 7-day week like us. Instead, they follow a 10-day week with the following days:

Day	Number Name of Day
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday
8	Kryptonday
9	Coluday
10	Daxamday

Here are the rules of the calendar:

- The calendar starts with Sunday always.
- It has only 296 days. After the 296th day, it goes back to Sunday.

You begin your journey on a Sunday and will reach after n. You have to tell on which day you will arrive when you reach there.

Input format:

- Contain a number n (0 < n)

Output format:

Print the name of the day you are arriving on

Sample Input

7

Sample Output

Kryptonday

Sample Input

1

Sample Output

Monday

Program:

```
#include<stdio.h>
int main()
{
    int n, day;
    scanf("%d", &n);
    if(n<296)
    day=n;
    else
    day=n-296;
    day%=10;
    day=day+1;
    day%=10;
    switch(day){
       case 1:
       printf("sunday");
       break;
       case 2:
       printf("Monday");
       break;
       case 3:
```

```
printf("Tuesday");
case 4:
printf("Wednesday");
break,
case 5
printf("Thursday");
break;
case 6:
printf("Friday");
break;
case 7
printf("saturday");
break;
case 8
printf("Kaptonday");
break;
case 9
printf("coluday");
break;
case 10:
printf("Daraday");
break;
}
```

Ex. No. : **Date :**

Stone Game-One Four

Problem Statement:

Alice and Bob are playing a game called "Stone Game". Stone game is a two-player game. Let N be the total number of stones. In each turn, a player can remove either one stone or four stones. The player who picks the last stone, wins. They follow the "Ladies First" norm. Hence Alice is always the one to make the first move. Your task is to find out whether Alice can win, if both play the game optimally.

Input Format
First line starts with T, which is the number of test cases. Each test case will contain N number of stones.

Output Format
Print "Yes" in the case Alice wins, else print "No".

Constraints 1<=T<=1000 1<=N<=10000

Sample Input
3
1
6
7

Sample Output
Yes
Yes
No

Program:

```
#include<stdio.h>
int main()
{
    int T, i=0, n, t;
    scanf("%d", &T);
    while(i<T)
    {
        scanf("%d", &n);
        t = n/4;
        if (t%2==0 && n%2==0)
        {
            printf("No\n");
        }
        else if (t%2==1 && n%2==1)
        {
            printf("No\n");
        }
        else
        {
            printf("Yes\n");
        }
        i++;
    }
}
```

Ex. No. : **Date :**

Holes in a Number

Problem Statement:

You are designing a poster which prints out numbers with a unique style applied to each of them. The styling is based on the number of closed paths or holes present in a given number.

The number of holes that each of the digits from 0 to 9 have are equal to the number of closed paths in the digit. Their values are:

1, 2, 3, 5, 7 = 0 holes.
0, 4, 6, 9 = 1 hole.
8 = 2 holes.

Given a number, you must determine the sum of the number of holes for all of its digits. For example, the number 819 has 3 holes.
Complete the program, it must return an integer denoting the total number of holes in num.

Constraints
$1 \leq num \leq 109$

Input Format For Custom Testing
There is one line of text containing a single integer num, the value to process.

Sample Input
630

Sample Output
2

Program:

```
#include<stdio.h>
int main()
{
   int a,b,n=0;
   scanf("%d",&a);
   while(a>0)
   {
         b=a%10;
         if(b==0||b==6||b==9||b==4)
         {
               n=n+1;
         }
         else if(b==8)
         {
               n=n+2;
         }
         a=a/10;
   }
   printf("%d",n);
}
```

Ex. No. : **Date :**

Philaland Coin

Problem Statement:

The problem solvers have found a new Island for coding and named it as Philaland. These smart people were given a task to make a purchase of items at the Island easier by distributing various coins with different values. Manish has come up with a solution that if we make coins category starting from $1 till the maximum price of the item present on Island, then we can purchase any item easily. He added the following example to prove his point.

Let's suppose the maximum price of an item is 5$ then we can make coins of {$1, $2, $3, $4, $5}to purchase any item ranging from $1 till $5.

Now Manisha, being a keen observer suggested that we could actually minimize the number of coins required and gave following distribution {$1, $2, $3}. According to him any item can be purchased one time ranging from $1 to $5. Everyone was impressed with both of them. Your task is to help Manisha come up with a minimum number of denominations for any arbitrary max price in Philaland.

Input Format
Contains an integer N denoting the maximum price of the item present on Philaland.

Output Format
Print a single line denoting the minimum number of denominations of coins required.

Constraints
1<=T<=100 1<=N<=5000

Sample Input 1:
10

Sample Output 1:
4

Program:

```
#include<stdio.h>
int main(){
    int n,r=0;
    scanf("%d",&n);
    while(n!=0)
    {
        n=n/2;
        r=r+1;
    }
    printf("%d",r);
}
```

Ex. No. : **Date :**

Number Count

Problem Statement:

A set of N numbers (separated by one space) is passed as input to the program. The program must identify the count of numbers where the number is odd number.

Input Format:
The first line will contain the N numbers separated by one space.

Boundary Conditions:
3 <= N <= 50
The value of the numbers can be from -99999999 to 99999999

Output Format:
The count of numbers where the numbers are odd numbers.

Sample Input:
5 10 15 20 25 30 35 40 45 50

Sample Output:
5

Program:

```
#include<stdio.h>
int main()
{
    int n, x=0;
    while(scanf("%d", &n)==1)
    {
        if(n%2!=0)
        {
            x++;
        }
    }
    printf("%d", x);
    return 0;
}
```

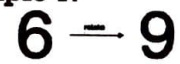
Ex. No. : **Date :**

Confusing Number

Problem Statement:

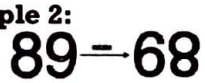
Given a number N, return true if and only if it is a *confusing number*, which satisfies the following condition:
We can rotate digits by 180 degrees to form new digits. When 0, 1, 6, 8, 9 are rotated 180 degrees, they become 0, 1, 9, 8, 6 respectively. When 2, 3, 4, 5 and 7 are rotated 180 degrees, they become invalid. A *confusing number* is a number that when rotated 180 degrees becomes a **different** number with each digit valid.

Example 1:



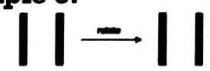
Input: 6
Output: true
Explanation: We get 9 after rotating 6, 9 is a valid number and 9 != 6.

Example 2:



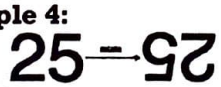
Input: 89
Output: true
Explanation: We get 68 after rotating 89, 86 is a valid number and 86 != 89.

Example 3:



Input: 11
Output: false
Explanation: We get 11 after rotating 11, 11 is a valid number but the value remains the same, thus 11 is not a confusing number.

Example 4:



Input: 25
Output: false
Explanation: We get an invalid number after rotating 25.
Note:

1. 0 <= N <= 10^9
2. After the rotation we can ignore leading zeros, for example if after rotation we have 0008 then this number is considered as just 8.

Program:

```
# include <stdio.h>
int main()
{
   int n, r, y=1;
   scanf("%d", &n);
   while (n!=0 && y==1) {
        r= n%10; n =n/10;
        if (r==2 || r==3 || r==4 || r==7) {
             y++; }}

   if (y==1) {
        printf ("true"); }
   else {
        printf("false"); }
}
```

Ex. No. : **Date :**

Nutrition Value

Problem Statement:

A nutritionist is labeling all the best power foods in the market. Every food item arranged in a single line, will have a value beginning from 1 and increasing by 1 for each, until all items have a value associated with them. An item's value is the same as the number of macronutrients it has. For example, food item with value 1 has 1 macronutrient, food item with value 2 has 2 macronutrients, and incrementing in this fashion.

The nutritionist has to recommend the best combination to patients, i.e. maximum total of macronutrients. However, the nutritionist must avoid prescribing a particular sum of macronutrients (an 'unhealthy' number), and this sum is known. The nutritionist chooses food items in the increasing order of their value. Compute the highest total of macronutrients that can be prescribed to a patient, without the sum matching the given 'unhealthy' number.

Here's an illustration: Given *4* food items (hence value: *1,2,3* and *4*), and the unhealthy sum being *6* macronutrients, on choosing items *1, 2, 3* -> the sum is *6,* which matches the 'unhealthy' sum. Hence, one of the three needs to be skipped. Thus, the best combination is from among:

- $2 + 3 + 4 = 9$
- $1 + 3 + 4 = 8$
- $1 + 2 + 4 = 7$

Since $2 + 3 + 4 = 9$, allows for maximum number of macronutrients, 9 is the right answer. Complete the code in the editor below. It must return an integer that represents the maximum total of macronutrients, modulo 1000000007 $(109 + 7)$.

It has the following:

n: an integer that denotes the number of food items

k: an integer that denotes the unhealthy number

Constraints

- $1 \leq n \leq 2 \times 109$
- $1 \leq k \leq 4 \times 1015$

Input Format For Custom Testing

The first line contains an integer, *n*, that denotes the number of food items. The second line contains an integer, *k*, that denotes the unhealthy number.

Sample Input 0

2
2

Sample Output 0

3

Program:

```
#include<stdio.h>
int main() {
    long long int n, t, i, sut = 0;
    scanf("%lld %lld", &n, &t);
    for (i=1; i<=n; i++) {
        sut = sut + i;
        if (sut == t) {
            sut = sut - i; } }
    printf("%lld ", sut %1000000007);
}
```

Ex. No. : **Date :**

Simple Chessboard

Problem Statement:

Write a program that prints a simple chessboard.

Input format:
The first line contains the number of inputs T.
The lines after that contain a different value for size of the chessboard

Output format:
Print a chessboard of dimensions size * size.
Print W for white spaces and B for black spaces.

Sample Input:
2
3
5

Sample Output:
WBW
BWB
WBW
WBWBW
BWBWB
WBWBW
BWBWB
WBWBW

Program:

```
#include<stdio.h>
int main() {
    int T, a, i=0, i1, i2, o;
    char c;
    scanf("%d", &T);
    while(i<T) {
        scanf("%d", &a);
        i1=0;
        while(i1<a)
        {
            o=1;
            i2=0;
            if(i1%2==0){
                o=0; }
            while(i2<a) {
                c='B';
                if(i2%2==0) {
                    c='W'; }
                printf("%c", c);
                i2++; }
            i1+=1;
            printf("\n");
        }
        i=i+1;
    }
}
```

Ex. No. : **Date :**

Print Our Own Chessboard

Problem Statement:

Let's print a chessboard!

Write a program that takes input:
The first line contains T, the number of test cases
Each test case contains an integer N and also the starting character of the chessboard

Output Format
Print the chessboard as per the given examples

Sample Input:
2
2 W
3 B

Sample Output:
WB
BW
BWB
WBW
BWB

Program:

```
#include<stdio.h>
int main() {
    int T, d, i, i1, i2, o, z;
    char c, s;
    scanf("%d", &T);
    for(i=0; i<T; i++) {
        scanf("%d %c", &d, &s);
        for(i1=0; i1<d; i1++) {
            z=(s=='w') ? 0:1;
            o=(i1%2==z) ? 0:1;
            for(i2=0; i2<d; i2++) {
                c=(i2%2==0) ? 'w' : 'B';
                printf("%c", c); }
            printf("\n"); } }
    return 0;
}
```

Ex. No. : **Date :**

Pattern Printing

Problem Statement:

Decode the logic and print the Pattern that corresponds to given input.

If N= 3 then pattern will be:
10203010011012
**4050809
****607
If N= 4, then pattern will be:
1020304017018019020
**50607014015016
****809012013
******10011

Constraints: 2 <= N <= 100
Input Format
First line contains T, the number of test cases, each test case contains a single integer N

Output Format
First line print Case #i where i is the test case number, In the subsequent line, print the pattern

Sample Input
3
3
4
5

Sample Output
Case #1
10203010011012
**4050809
****607
Case #2
1020304017018019020
**50607014015016
****809012013
******10011
Case #3
102030405026027028029030
**6070809022023024025
****10011012019020021
******13014017018
********15016

Program:

```
#include<stdio.h>
int main() {
    int n, v, p3, c, in, i, i1, i2, t, ti;
    scanf("%d", &t);
    for(ti=0; ti<t; ti++){
        v=0;
        scanf("%d", &n);
        printf("Case %d\n", ti+1);
        for(i=0; i<n; i++) {
            c=0;
            if(i>0) {
                for (i1=0; i1<i; i1++) {
                    printf(" * *"); }}
            for(i1=i; i1<n; i1++){
                if (i>0) c++;
                printf("%d0", ++v);
            }
            if(i=0) {
                p3=v+(v*(v-1))+1;
                in=p3; }
            in=in-c;
            p3=in;
            for(i2=i; i2<n; i2++) {
                printf("%d", p3++);
                if (i2!=n-1) printf("0"); }
            printf("\n"); }}
```

Ex. No. : **Date :**

Armstrong Number

Problem Statement:

The k-digit number N is an Armstrong number if and only if the k-th power of each digit sums to N.

Given a positive integer N, return true if and only if it is an Armstrong number.

Note: 1 <= N <= 10^8

Hint: 153 is a 3-digit number, and 153 = 1^3 + 5^3 + 3^3.

Sample Input:
153

Sample Output:
true

Sample **Input:**
123

Sample **Output:**
false

Sample **Input:**
1634

Sample **Output:**
true

Program:

```
#include<stdio.h>
#include<math.h>
int main() {
   int n;
   scanf("%d", &n);
   int x=0, n2=n;
   while(n2!=0) {
        x++;
        n2 = n2/10; }
   int sum=0;
   int n3=n, n4;
   while(n3!=0){
      n4 = n3%10;
      sum= sum+ pow(n4, x);
      n3 = n3/10; }
   if(n==sum) {
      printf("true"); }
   else {
      printf("false"); }
   return 0;
}
```