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

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

    }
    return 0;
}
```

	Input	Expected	Got	
✓	3 1 6 7	Yes Yes No	Yes Yes No	✓

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, and 7 = 0 holes.

0, 4, 6, and 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 must return an integer denoting the total number of holes in num.

```c
#include <stdio.h>
int main()
{
    int n,temp,hole=0;
    scanf("%d", &n);
    while(n!=0)
    {


        temp= n%10;
        switch(temp)
        {
            case 1:
            case 2:
            case 3:
            case 5:
            case 7:
             hole=hole +0;
             break;
             break;
            case 0:
            case 4:
            case 6:
            case 9:
                hole=hole +1;
                break;
            case 8:
                hole=hole+2;
                break;


        }
        n=n/10;
        n=n/10;
    }
    printf("%d", hole);
    return 0;
}
```

	Input	Expected	Got	
✓	630	2	2	✓
✓	1288	4	4	✓

Passed all tests! ✓

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.

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

	Input	Expected	Got	
✓	10	4	4	✓
✓	5	3	3	✓
✓	20	5	5	✓
✓	500	9	9	✓
✓	1000	10	10	✓

Question **1**

Correct

Marked out of 3.00

⚑ Flag question

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.

```c
#include <stdio.h>

int main()
{
    int n,x=0;
    while((scanf("%d", &n)==1))
    {
        if(n%2!=0)
        {
            x++;
        }
    }
    printf("%d", x);
    return 0;
}
```

	Input	Expected	Got	
✓	5 10 15 20 25 30 35 40 45 50	5	5	✓

Passed all tests! ✓

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:

6 -> 9

Input: 6

Output: true

Explanation:

We get 9 after rotating 6, 9 is a valid number and 9!=6.

```c
1   #include <stdio.h>
2   int main()
3   {
4       int n,k=1,x;
5       scanf("%d", &n);
6       while(n!=0 && k==1)
7       {
8           x=n%10;
9           n=n/10;
10          if(x==2 || x==3 || x==4 || x==7)
11          {
12              k++;
13          }
14      }
15      if(k==1)
16      printf("true");
17      else
18      printf("false");
19      return 0;
20  }
21
```

	Input	Expected	Got	
✓	6	true	true	✓
✓	89	true	true	✓
✓	25	false	false	✓

Passed all tests! ✓

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)*.

```
1   #include <stdio.h>
2   int main()
3   {
4       long long int n, u,s=0;
5       scanf("%lld",&n);
6       scanf("%lld", &u);
7       for(int i=1; i<=n;i++)
8       {
9           s+=i;
10          if(s==u)
11          {
12              s=s-1;
13          }
14      }
15
16
17      printf("%lld",s%1000000007);
18      return 0;
19  }
```

	Input	Expected	Got	
✓	2 2	3	3	✓
✓	2 1	2	2	✓
✓	3 3	5	5	✓

Passed all tests! ✓

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 values for size of the chessboard

Output format:

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

Input:

2

3

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

                }
                printf("%c", c);
                i2++;
            }
            i1+=1;
            printf("\n");
        }
        i=i+1;
    }
    return 0;
}
```

	Input	Expected	Got	
✓	2	WBW	WBW	✓
	3	BWB	BWB	
	5	WBW	WBW	
		WBWBW	WBWBW	
		BWBWB	BWBWB	
		WBWBW	WBWBW	
		BWBWB	BWBWB	
		WBWBW	WBWBW	

Passed all tests! ✓

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 / Output

Input:

2

2 W

3 B

Output:

WB

BW

```
 1  #include <stdio.h>
 2  int main()
 3  {
 4      int t,d,i,i1,i2,o,z;
 5      char c,s;
 6      scanf("%d", &t);
 7      for(i=0;i<t;i++)
 8      {
 9          scanf("%d %c", &d,&s);
10          for(i1=0;i1<d;i1++)
11          {
12              z=(s=='W')?0:1;
13              o=(i1%2==z)?0:1;
14              for(i2=0;i2<d;i2++)
15              {
16                  c=(i2%2==o)?'W':'B';
17                  printf("%c",c);
18              }
19              printf("\n");
20          }
21      }
22      return 0;
23  }
```

	Input	Expected	Got	
✓	2	WB	WB	✓
	2 W	BW	BW	
	3 B	BWB	BWB	
		WBW	WBW	
		BWB	BWB	

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

```c
#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");
```

```
35            |     |         }printf("\n");
36            |     }
37            |   }
38        }
39        return 0;
40    }
```

	Input	Expected	Got	
✓	3 3 4 5	Case #1 10203010011012 **4050809 ****607 Case #2 1020304017018019020 **50607014015016 ****809012013 ******10011 Case #3 102030405026027028029030 **6070809022023024025 ****10011012019020021 ******13014017018 ********15016	Case #1 10203010011012 **4050809 ****607 Case #2 1020304017018019020 **50607014015016 ****809012013 ******10011 Case #3 102030405026027028029030 **6070809022023024025 ****10011012019020021 ******13014017018 ********15016	✓

Passed all tests! ✓