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 and Output

Input

3
1
6
7

Output

Yes
Yes
No

Answer: (penalty regime: 0 %)

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

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

Passed all tests! ✓

Complete the program, it must must return an integer denoting the total number of holes in num.

Constraints

1 ≤ num ≤ 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

Explanation

Add the holes count for each digit, 6, 3 and 0. Return 1 + 0 + 1 = 2.

Sample Case 1

Sample Input

1288

Sample Output

4

Explanation

Add the holes count for each digit, 1, 2, 8, 8. Return 0 + 0 + 2 + 2 = 4.

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main(){
    int b,a,holes=0;
    scanf("%d",&b);
    while(b>0){
        a =b%10;
        if(a==1||a==2||a==3||a==5||a==7){
            holes=holes+0;
        }
    else if(a==8){
        holes=holes+2;
    }
    else{
        holes=holes+1;
    }
    b=b/10;
    }

    printf("%d",holes);
}
```

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

Passed all tests! ✓

Question 3
Correct

The problem solvers have found a new Island for coding and named it as Philaland. These smart people were given a task

10

Sample Output 1:

4

Sample Input 2:

5

Sample Output 2:

3

Explanation:

For test case 1, N=10.

According to Manish {$1, $2, $3,... $10} must be distributed.

But as per Manisha only {$1, $2, $3, $4} coins are enough to purchase any item ranging from $1 to $10. Hence minimum is 4. Likewise denominations could also be {$1, $2, $3, $5}. Hence answer is still 4.

For test case 2, N=5.

According to Manish {$1, $2, $3, $4, $5} must be distributed.

But as per Manisha only {$1, $2, $3} coins are enough to purchase any item ranging from $1 to $5. Hence minimum is 3. Likewise, denominations could also be {$1, $2, $4}. Hence answer is still 3.

Answer: (penalty regime: 0 %)

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

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

Passed all tests! ✓

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.

Example Input / Output 1:

Input:

5 10 15 20 25 30 35 40 45 50

Output:

5

Explanation:

The numbers meeting the criteria are 5, 15, 25, 35, 45.

Answer: (penalty regime: 0 %)

```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:

Question 2

Correct

Marked out of 5.00

Flag question

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.

Example 2:

89 -> 68

Input: 89

Output: true

Explanation:

We get 68 after rotating 89, 86 is a valid number and 86!=89.

Example 3:

11 -> 11

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.

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.

Answer: (penalty regime: 0 %)

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

    }
}
```

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

Passed all tests! ✓

Question 3

Correct

Marked out of 7.00

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 following sequence of $n = 2$ food items:

1. Item 1 has 1 macronutrients.
2. $1 + 2 = 3$; observe that this is the max total, and having avoided having exactly $k = 2$ macronutrients.

Sample Input 1

2
1

Sample Output 1

2

Explanation 1

1. Cannot use item *1* because $k = 1$ and $sum \equiv k$ has to be avoided at any time.
2. Hence, max total is achieved by $sum = 0 + 2 = 2$.

Sample Case 2

Sample Input For Custom Testing

Sample Input 2

3
3

Sample Output 2

5

Explanation 2

$2 + 3 = 5$, *is* the best case for maximum nutrients.

Answer: (penalty regime: 0 %)

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

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

Passed all tests! ✓

Finish review