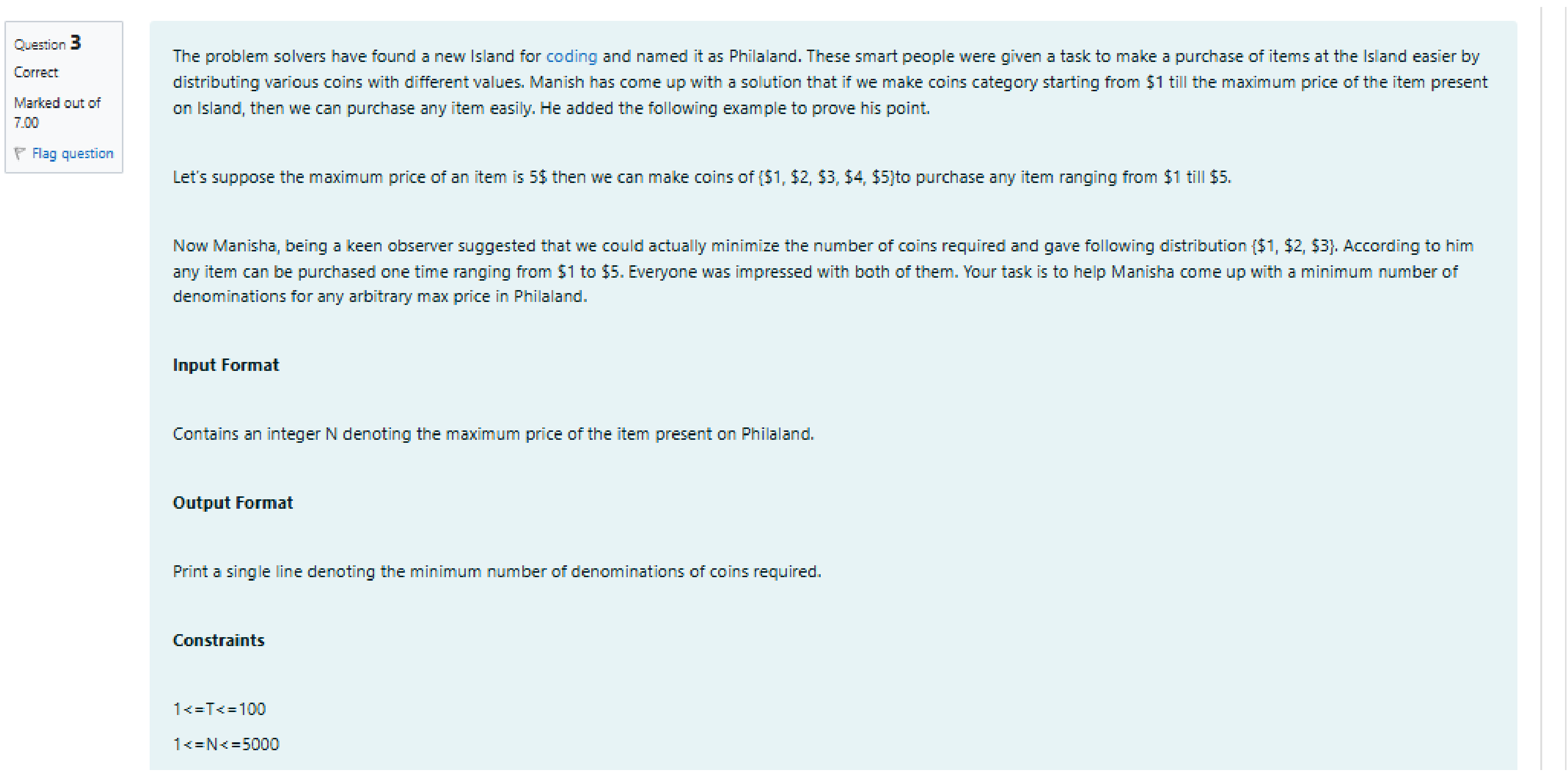


Question **3**

Correct

Marked out of 7.00

Flag question

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.

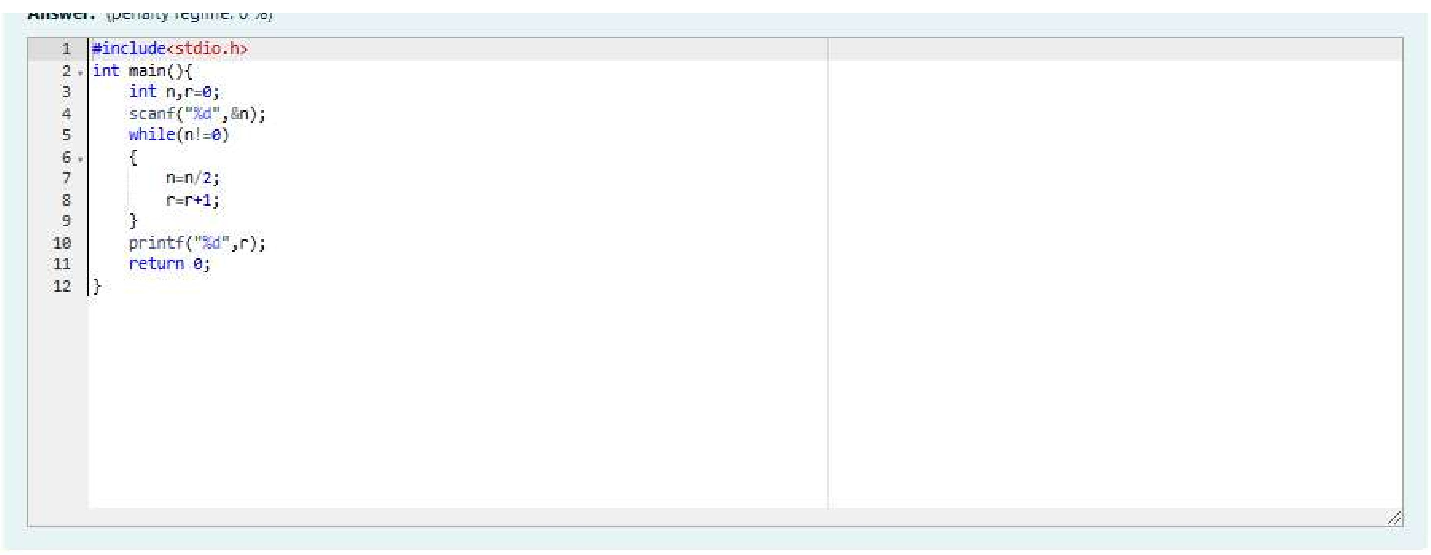
Output Format

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

Constraints

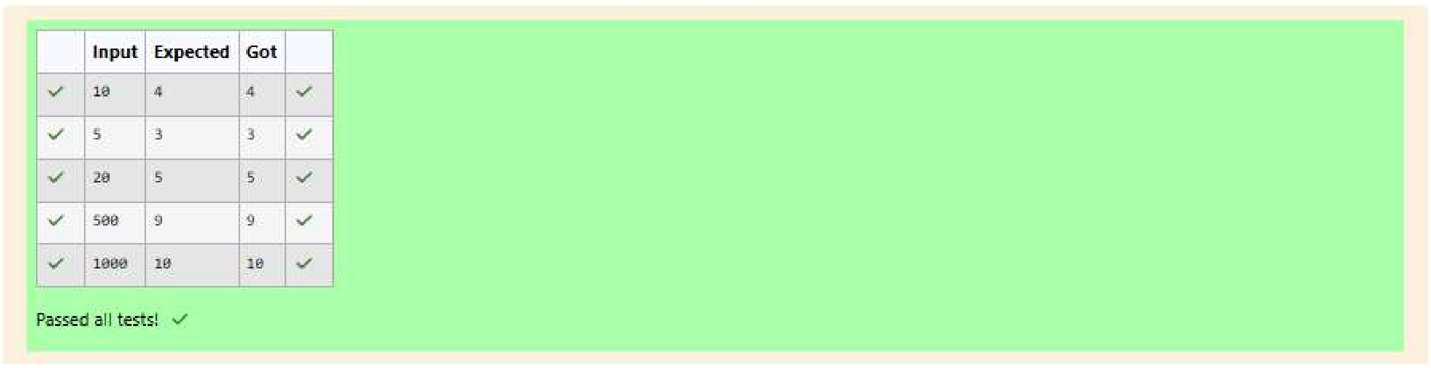
1<=T<=100

1<=N<=5000



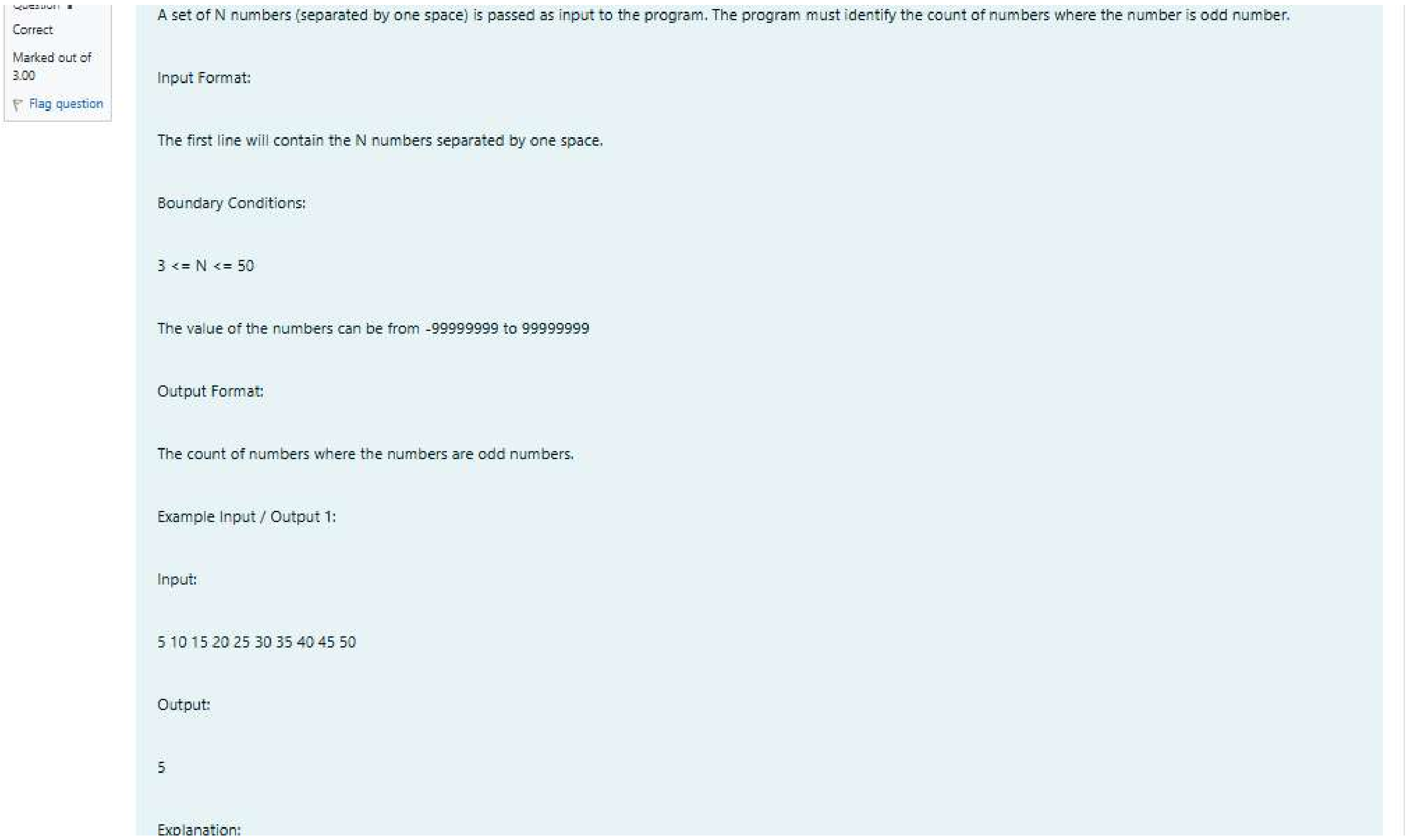
Answer: (penalty regime: 0 %)

```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	✓

Passed all tests! ✓



Question

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.

Example Input / Output 1:

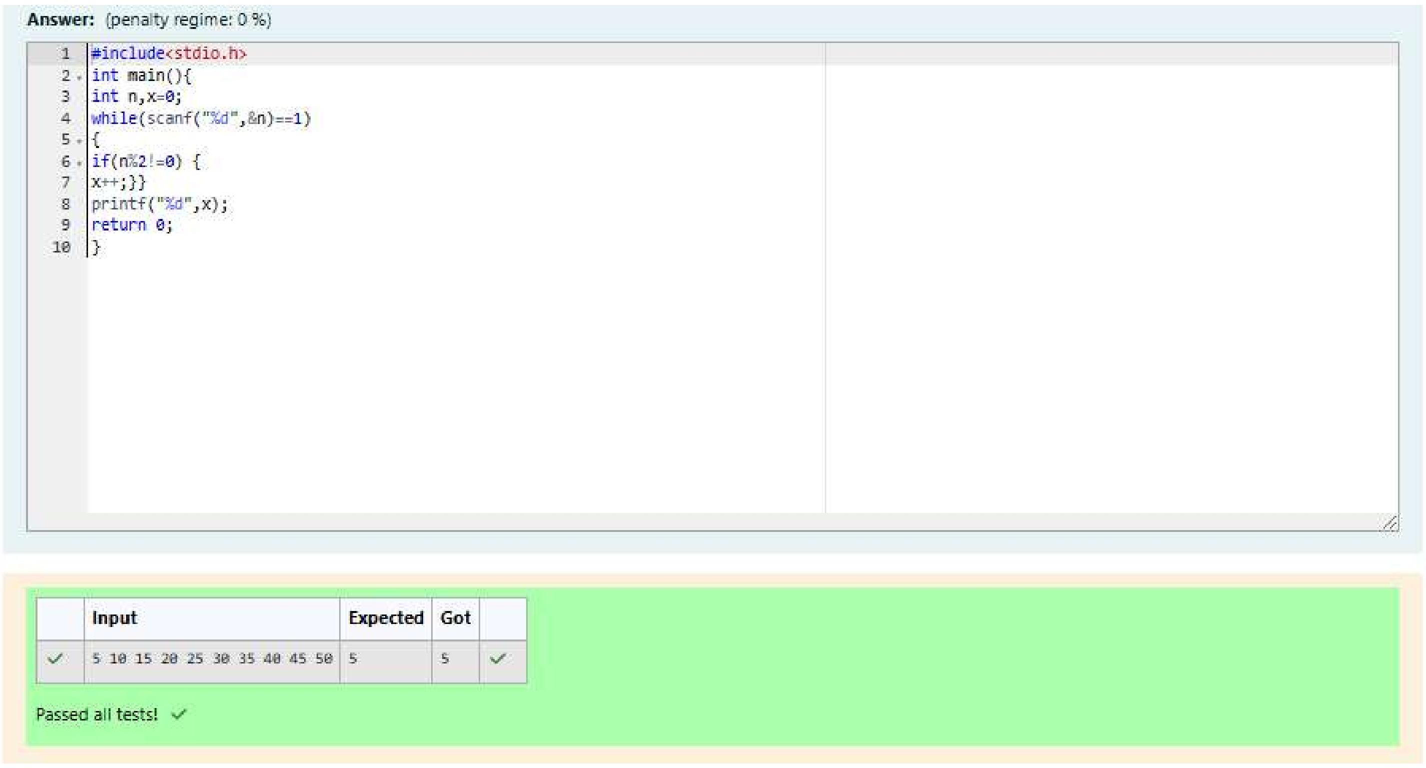
Input:

```
5 10 15 20 25 30 35 40 45 50
```

Output:

```
5
```

Explanation:



Answer: (penalty regime: 0 %)

```
#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! ✓

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:

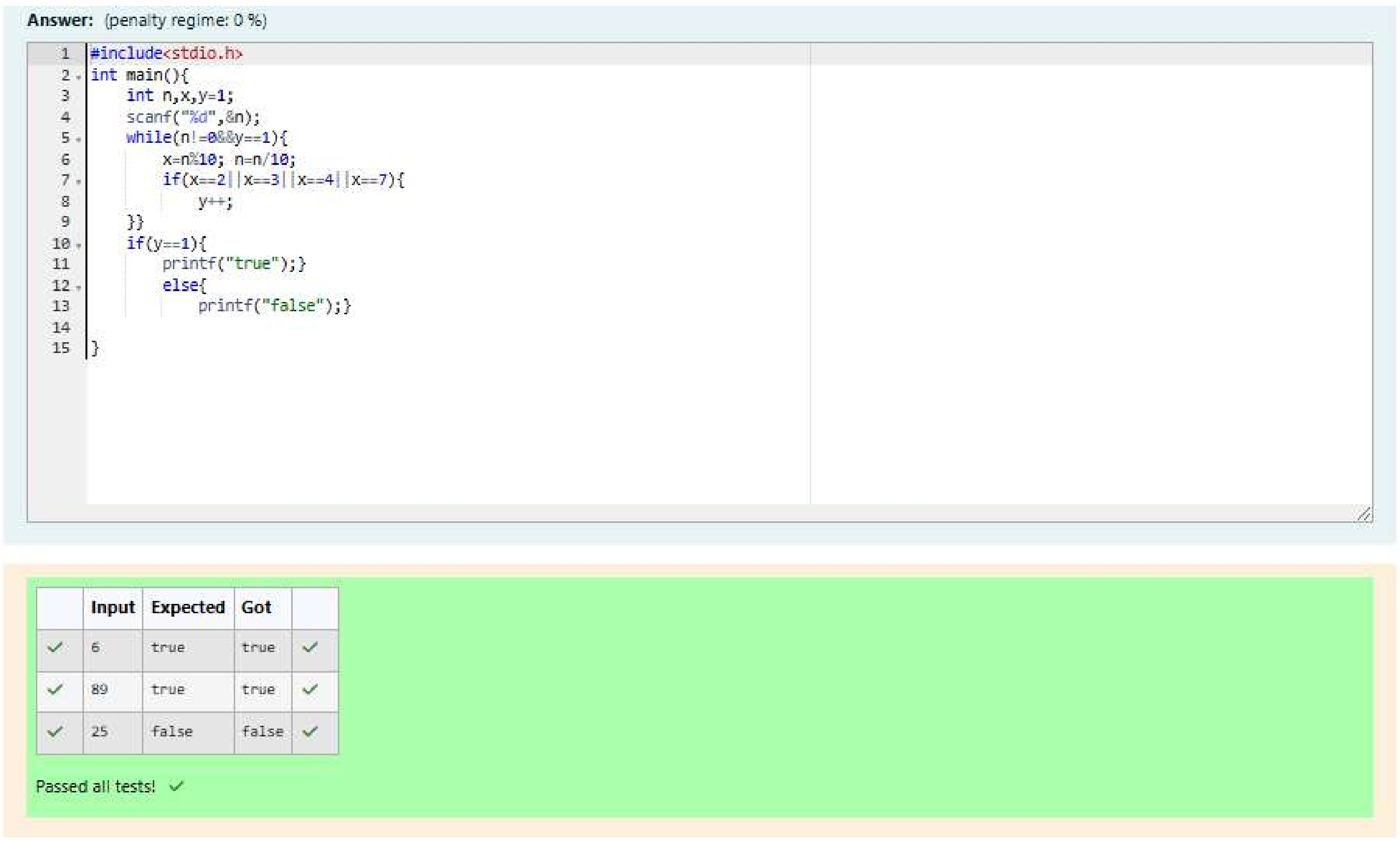
11 -> 11

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.

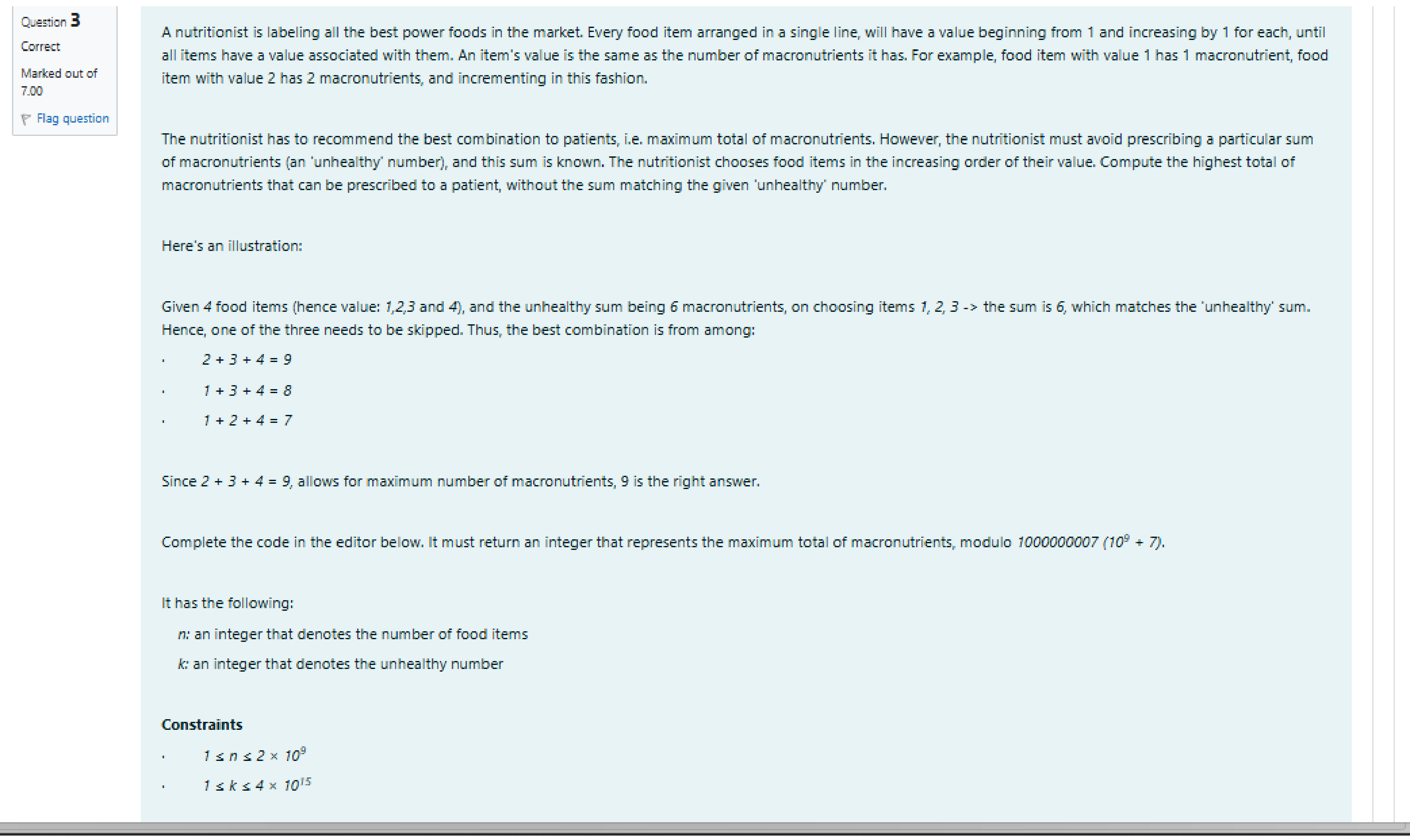


Answer: (penalty regime: 0 %)

```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

Flag question

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* ($10^9 + 7$).

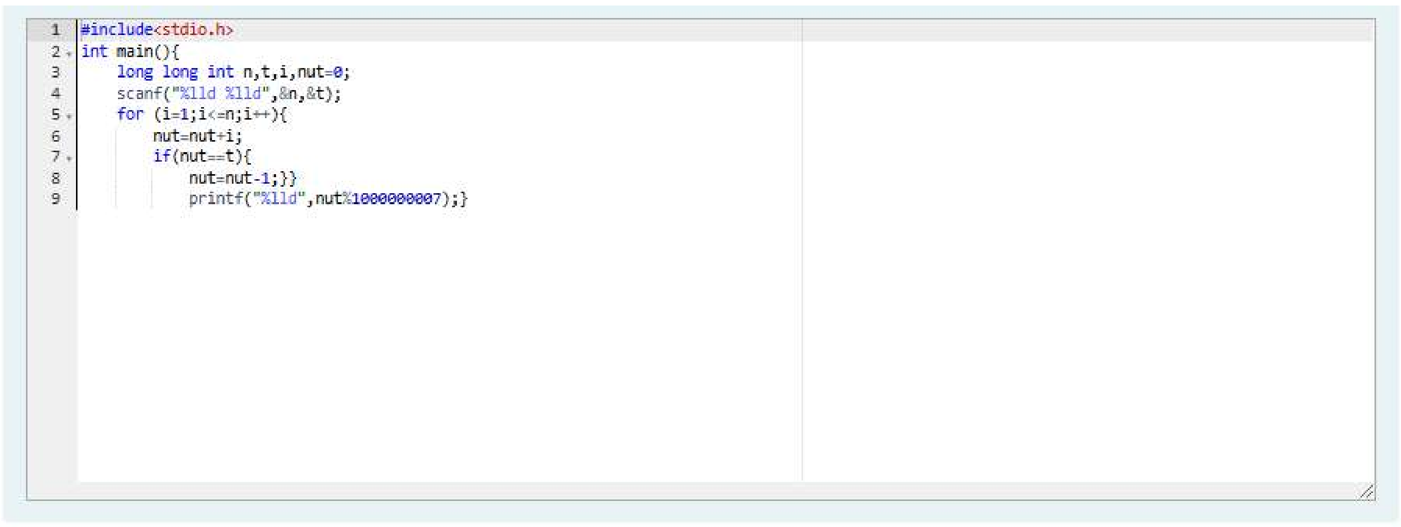
It has the following:

n: an integer that denotes the number of food items

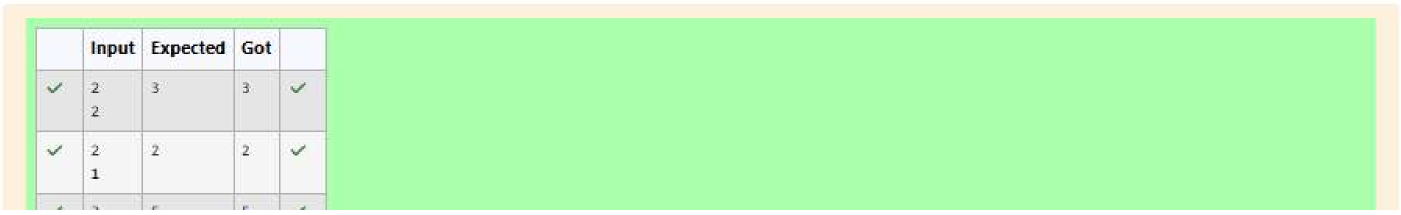
k: an integer that denotes the unhealthy number

Constraints

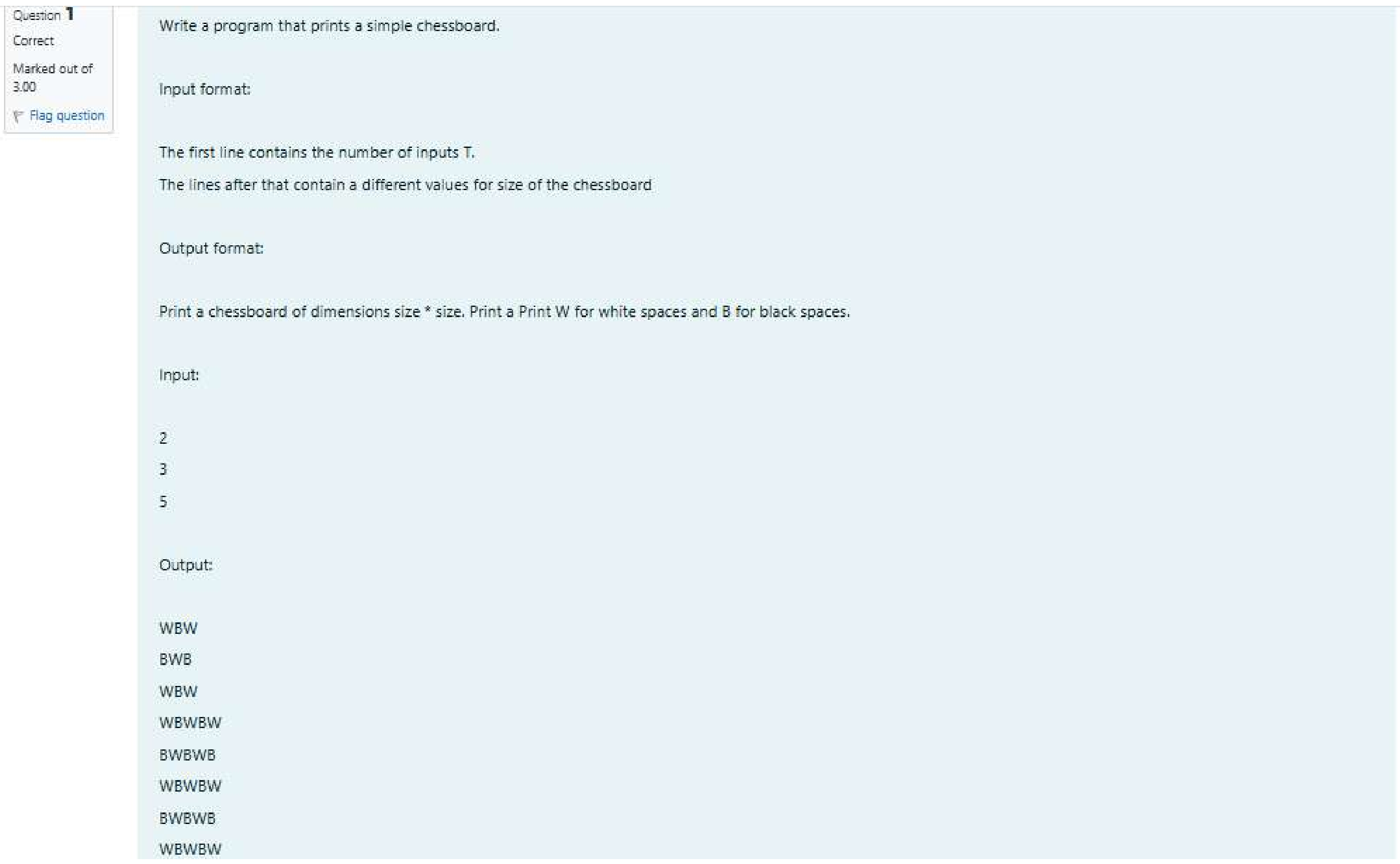
- $1 \le n \le 2 \times 10^9$
- $1 \le k \le 4 \times 10^{15}$



```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	✓



Question 1

Correct

Marked out of 3.00

Flag question

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:

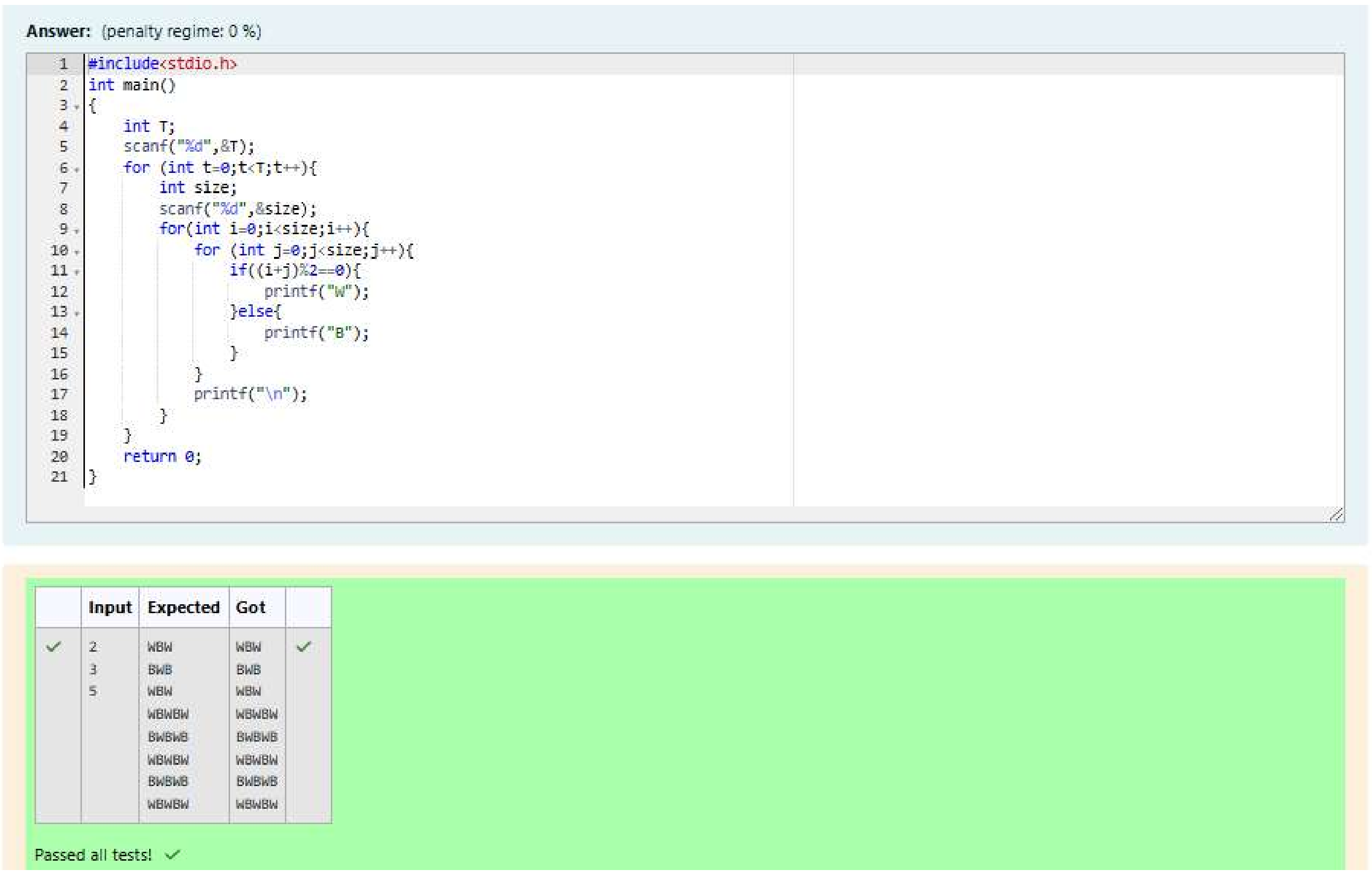
Print a chessboard of dimensions size * size. Print a Print W for white spaces and B for black spaces.

Input:

```
2
3
5
```

Output:

```
WBW
BWB
WBW
WBWBW
BWBWB
WBWBW
BWBWB
WBWBW
```

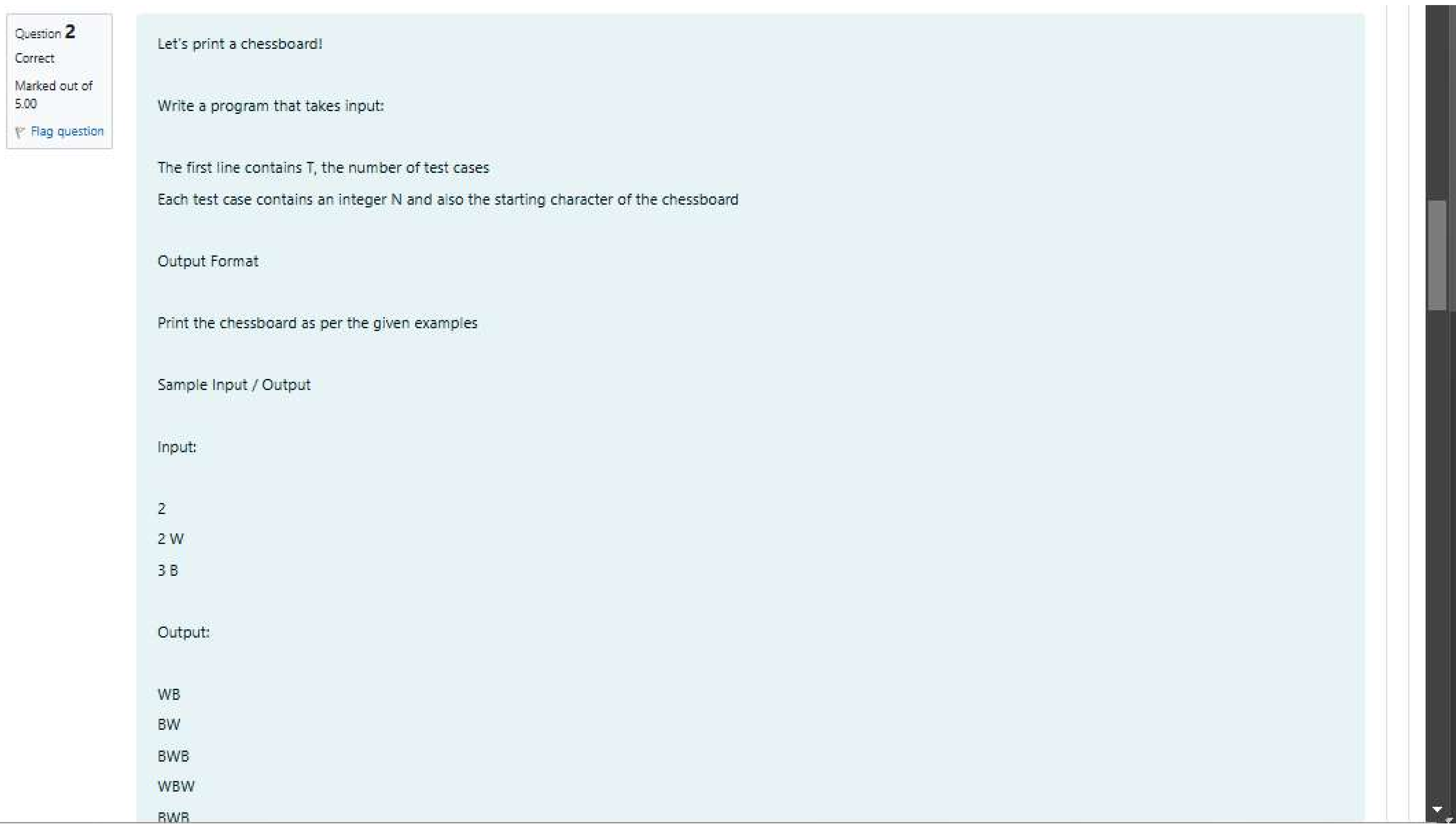


Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    int T;
    scanf("%d",&T);
    for (int t=0;t<T;t++){
        int size;
        scanf("%d",&size);
        for(int i=0;i<size;i++){
            for (int j=0;j<size;j++){
                if((i+j)%2==0){
                    printf("W");
                }else{
                    printf("B");
                }
            }
            printf("\n");
        }
    }
    return 0;
}
```

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

Passed all tests! ✓



Question **2**

Correct

Marked out of 5.00

Flag question

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
BWB
WBW
BWB
```

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    int T;
    scanf("%d",&T);
    for(int t=0;t<T;t++){
        int N;
        char start;
        scanf("%d %c",&N,&start);
        char alt=(start=='W')?'B':'W';
        for(int i=0;i<N;i++){
            for(int j=0;j<N;j++){
                if((i+j)%2==0){
                    printf("%c",start);
                }else{
                    printf("%c",alt);
                }
            }
            printf("\n");
        }
    }
    return 0;
}
```

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

Passed all tests! ✓

Question **3**

Correct

Marked out of 7.00

Flag question

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
```

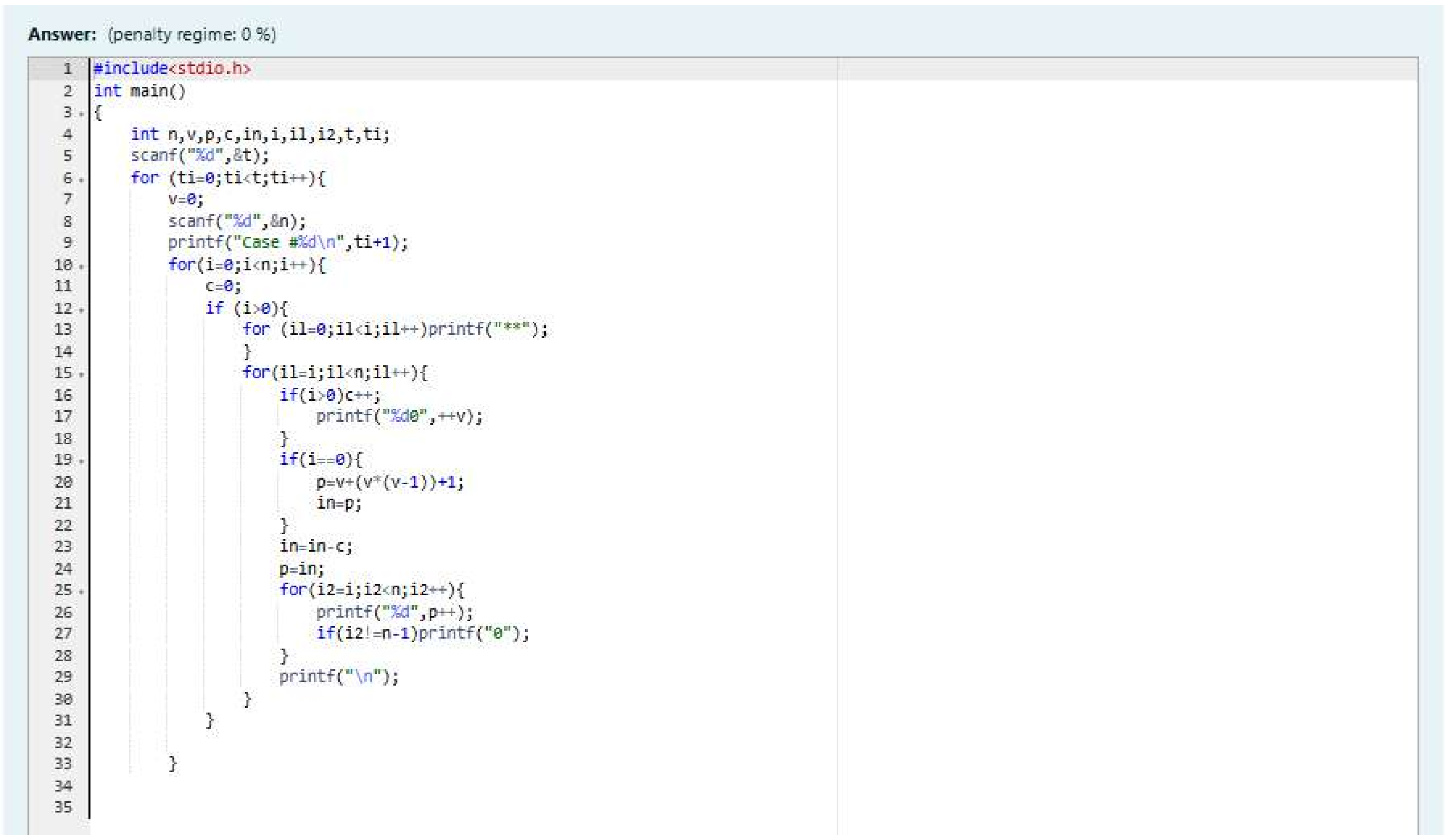
Constraints

2 <= N <= 100

Input Format

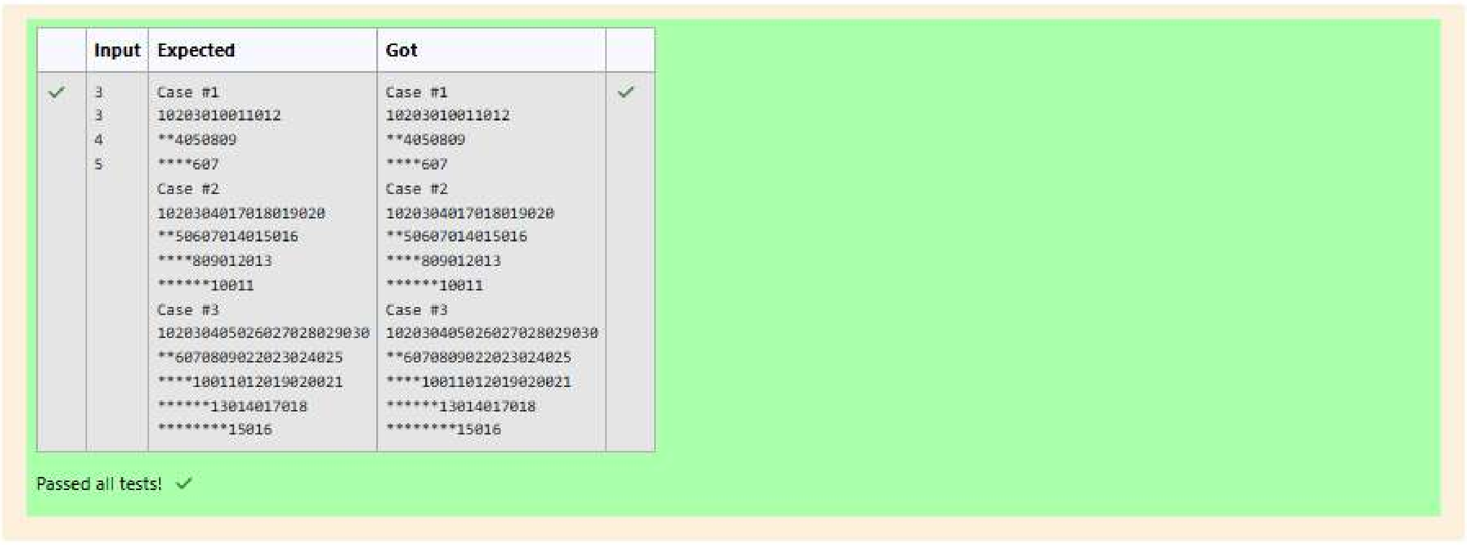
First line contains T, the number of test cases

Each test case contains a single integer N



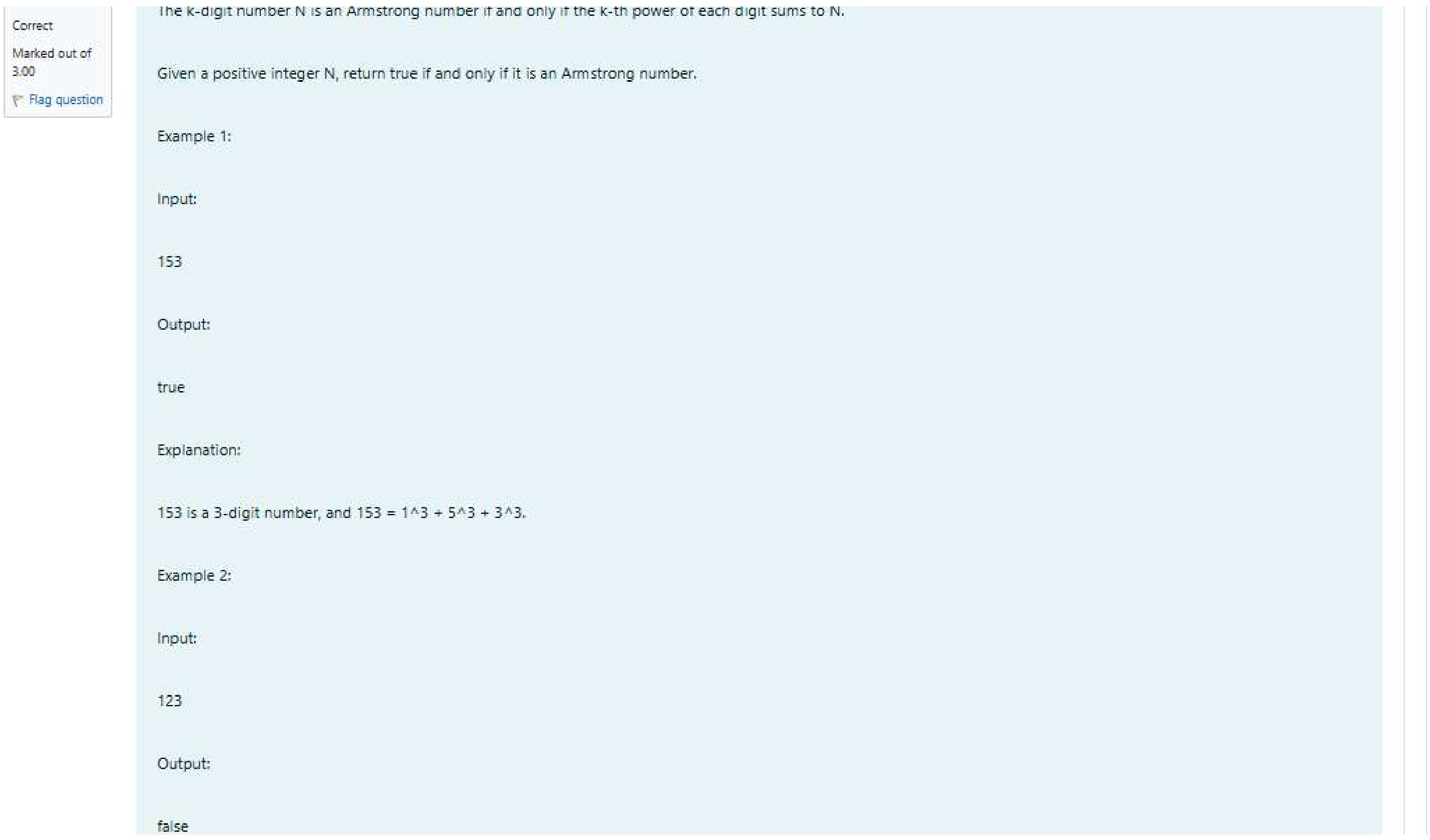
Answer: (penalty regime: 0 %)

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

        }


```



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

Passed all tests! ✓



Correct

Marked out of 3.00

Flag question

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.

Example 1:

Input:

153

Output:

true

Explanation:

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

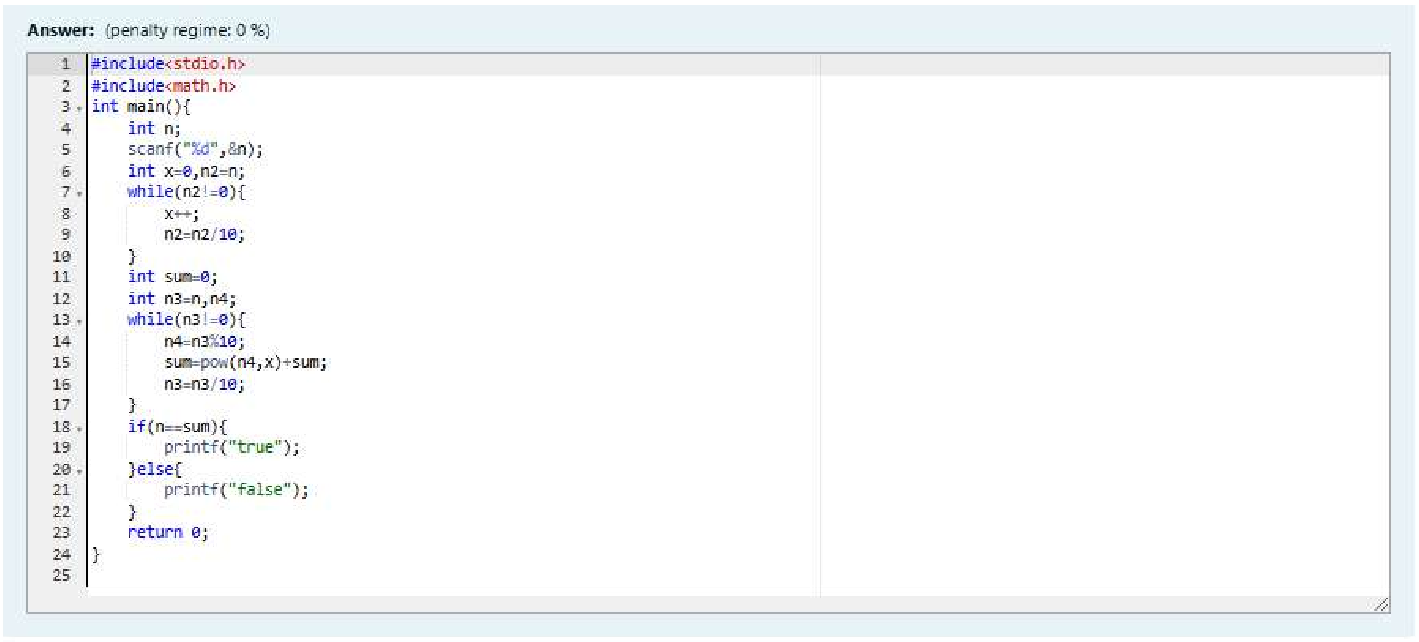
Example 2:

Input:

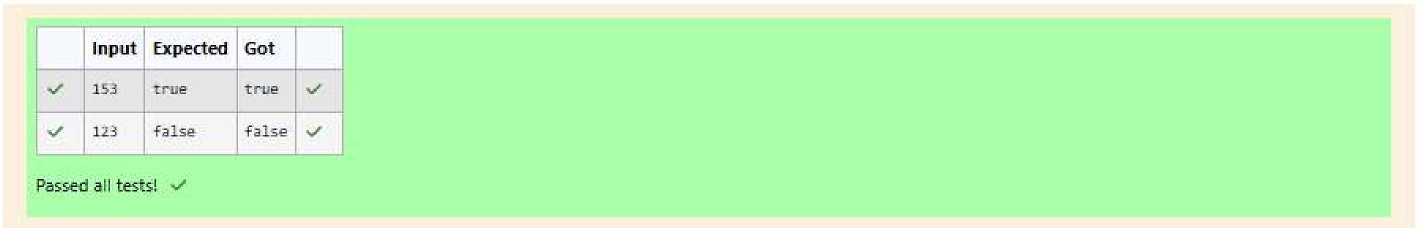
123

Output:

false

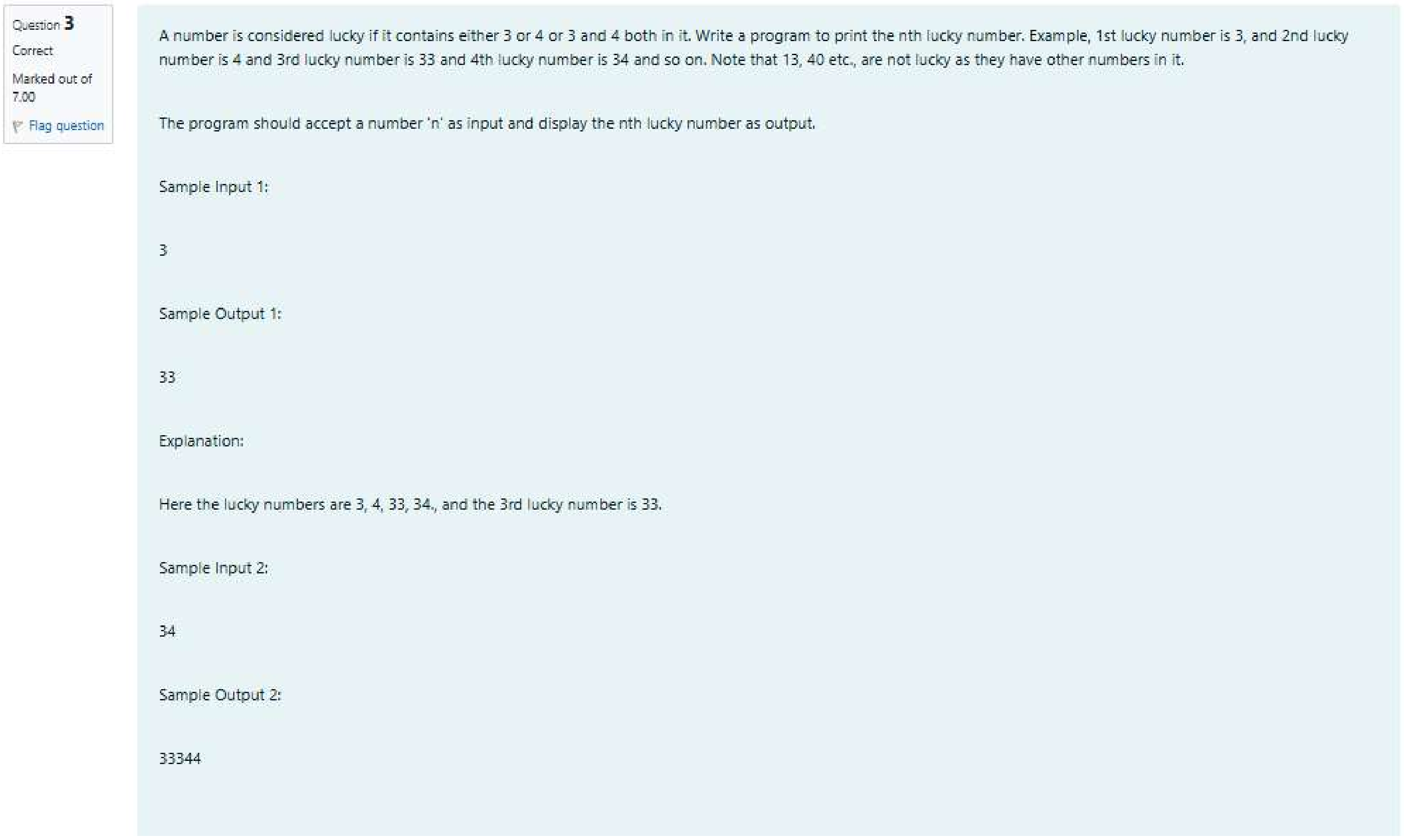

Answer: (penalty regime: 0 %)

```c
#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=pow(n4,x)+sum;
        n3=n3/10;
    }
    if(n==sum){
        printf("true");
    }else{
        printf("false");
    }
    return 0;
}
```



	Input	Expected	Got	
✓	153	true	true	✓
✓	123	false	false	✓

Passed all tests! ✓


Question **3**

Correct

Marked out of 7.00

Flag question

A number is considered lucky if it contains either 3 or 4 or 3 and 4 both in it. Write a program to print the nth lucky number. Example, 1st lucky number is 3, and 2nd lucky number is 4 and 3rd lucky number is 33 and 4th lucky number is 34 and so on. Note that 13, 40 etc., are not lucky as they have other numbers in it.

The program should accept a number 'n' as input and display the nth lucky number as output.

Sample Input 1:

3

Sample Output 1:

33

Explanation:

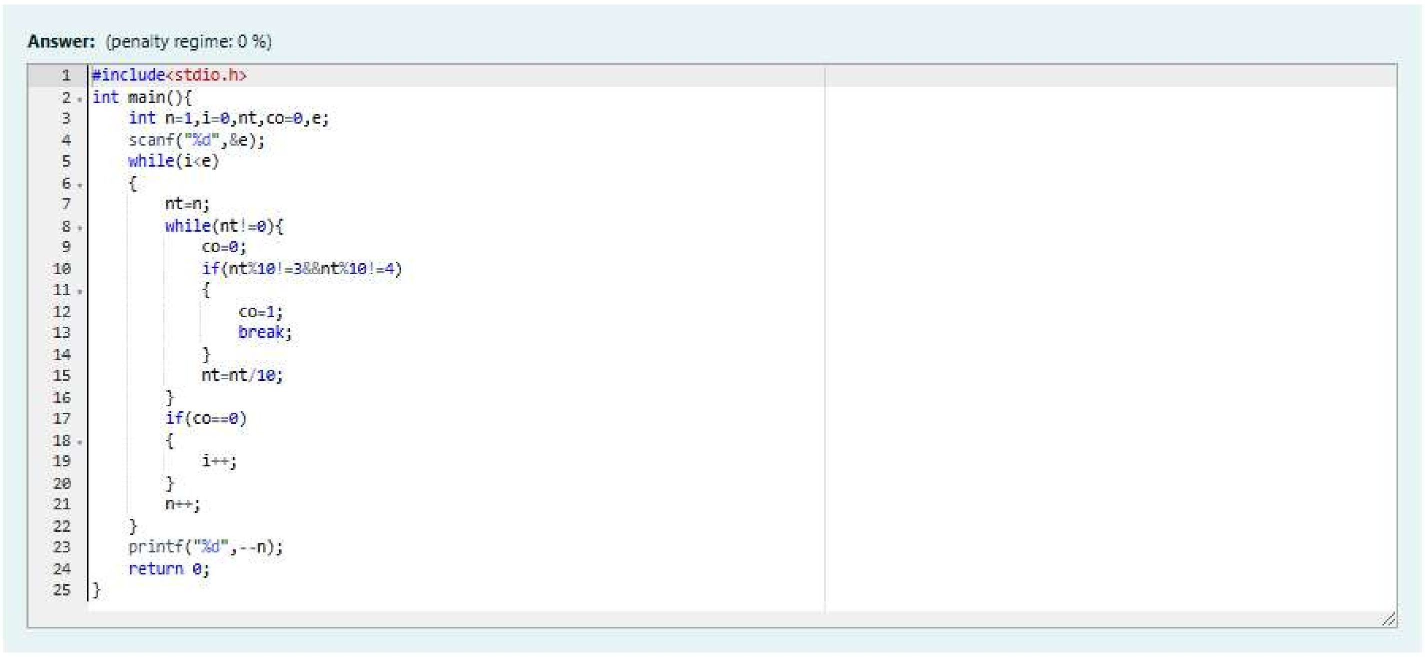
Here the lucky numbers are 3, 4, 33, 34., and the 3rd lucky number is 33.

Sample Input 2:

34

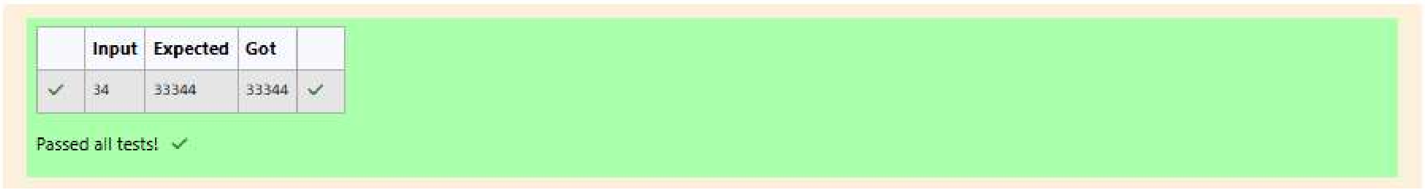
Sample Output 2:

33344



Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main(){
    int n=1,i=0,nt,co=0,e;
    scanf("%d",&e);
    while(i<e)
    {
        nt=n;
        while(nt!=0){
            co=0;
            if(nt%10!=3&&nt%10!=4)
            {
                co=1;
                break;
            }
            nt=nt/10;
        }
        if(co==0)
        {
            i++;
        }
        n++;
    }
    printf("%d",--n);
    return 0;
}
```



	Input	Expected	Got	
✓	34	33344	33344	✓

Passed all tests! ✓