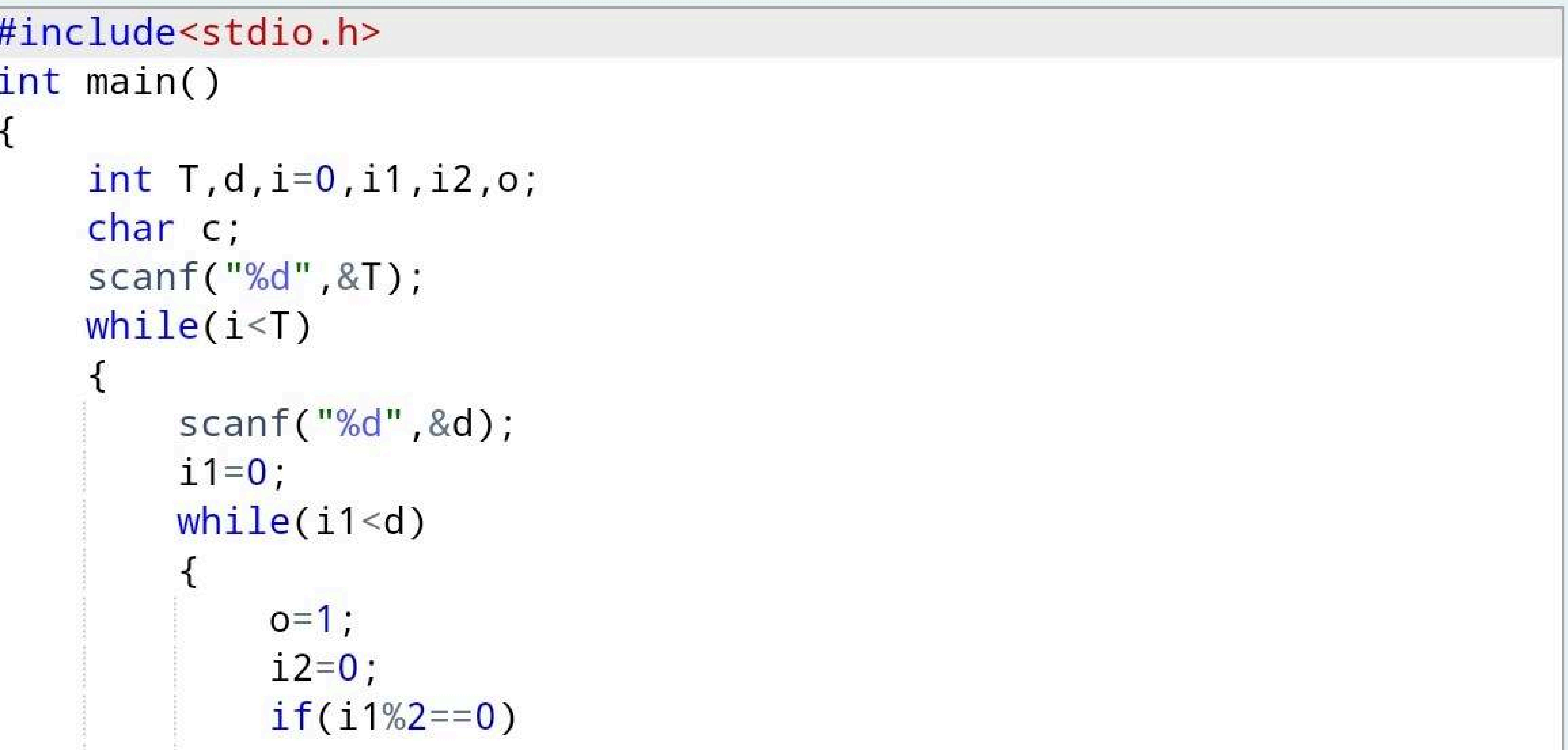
Write a program that prints a simple chessboard.

Input format:

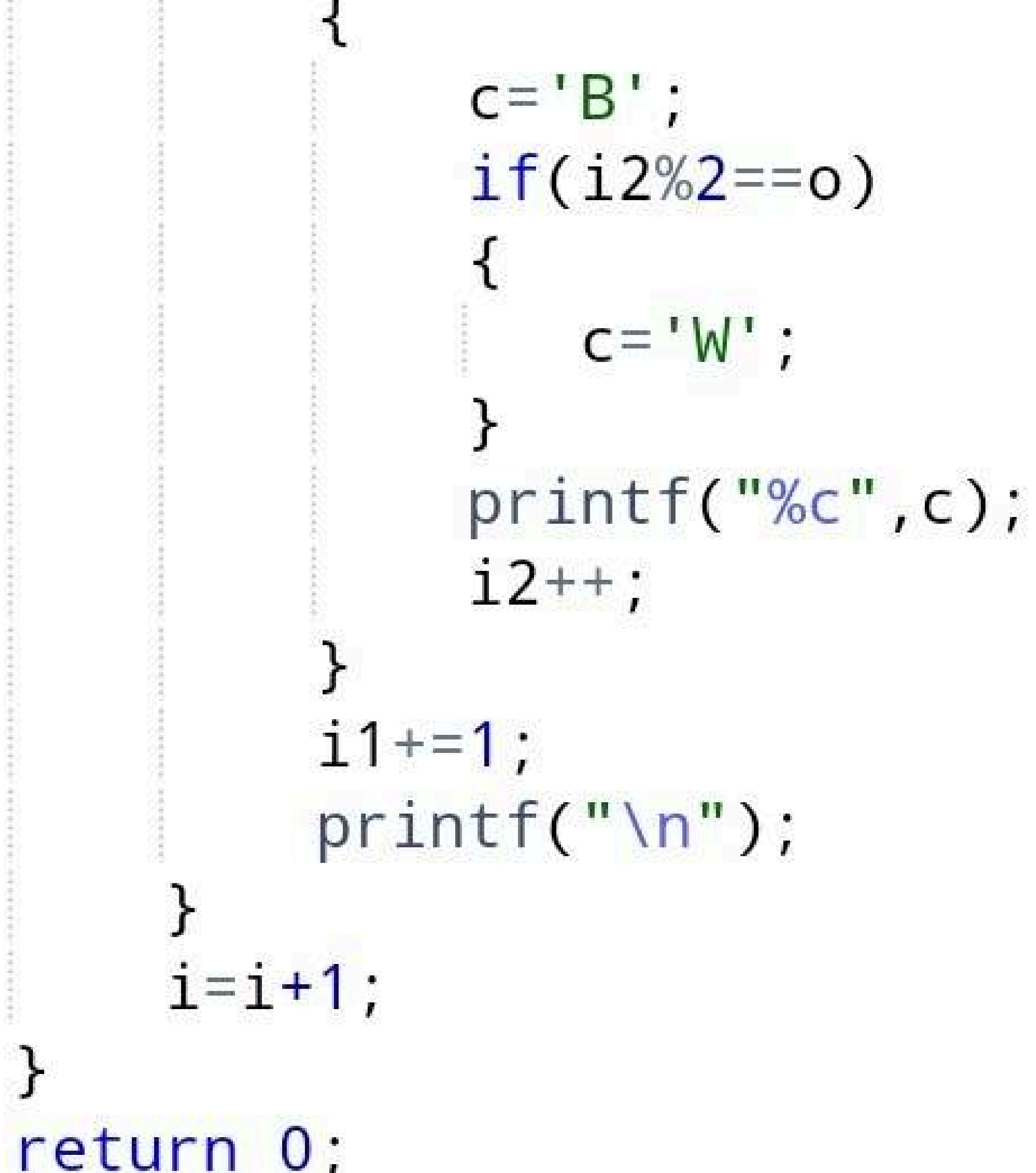
The first line contains the number of inputs T.

The lines after that contain a different values for size of the chessboard

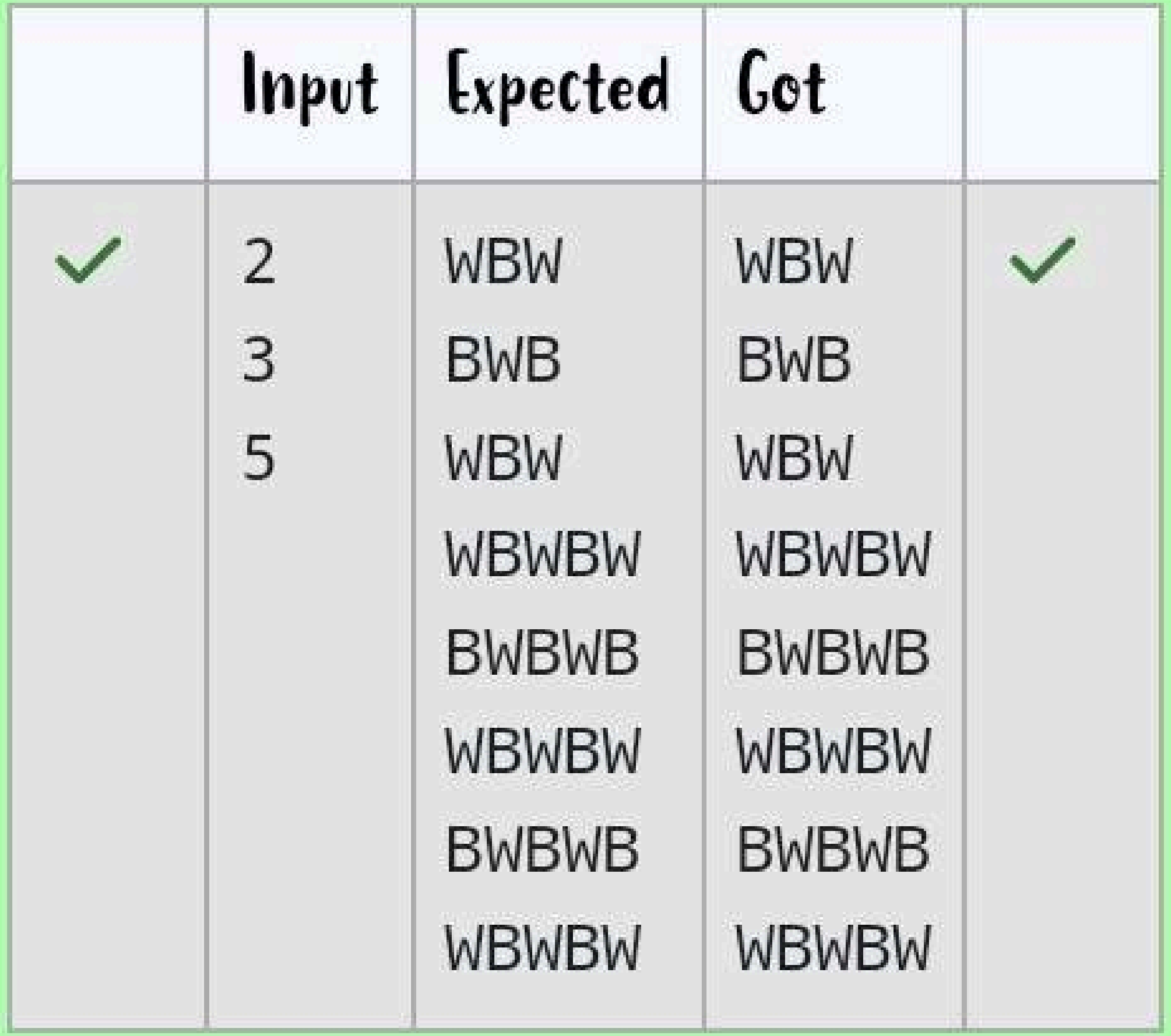
Output format:



```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)
```



```
        {
            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 3 5	WBW BWB WBW WBWBW BWBWB WBWBW BWBWB WBWBW	WBW BWB WBW WBWBW BWBWB WBWBW BWBWB WBWBW	✓

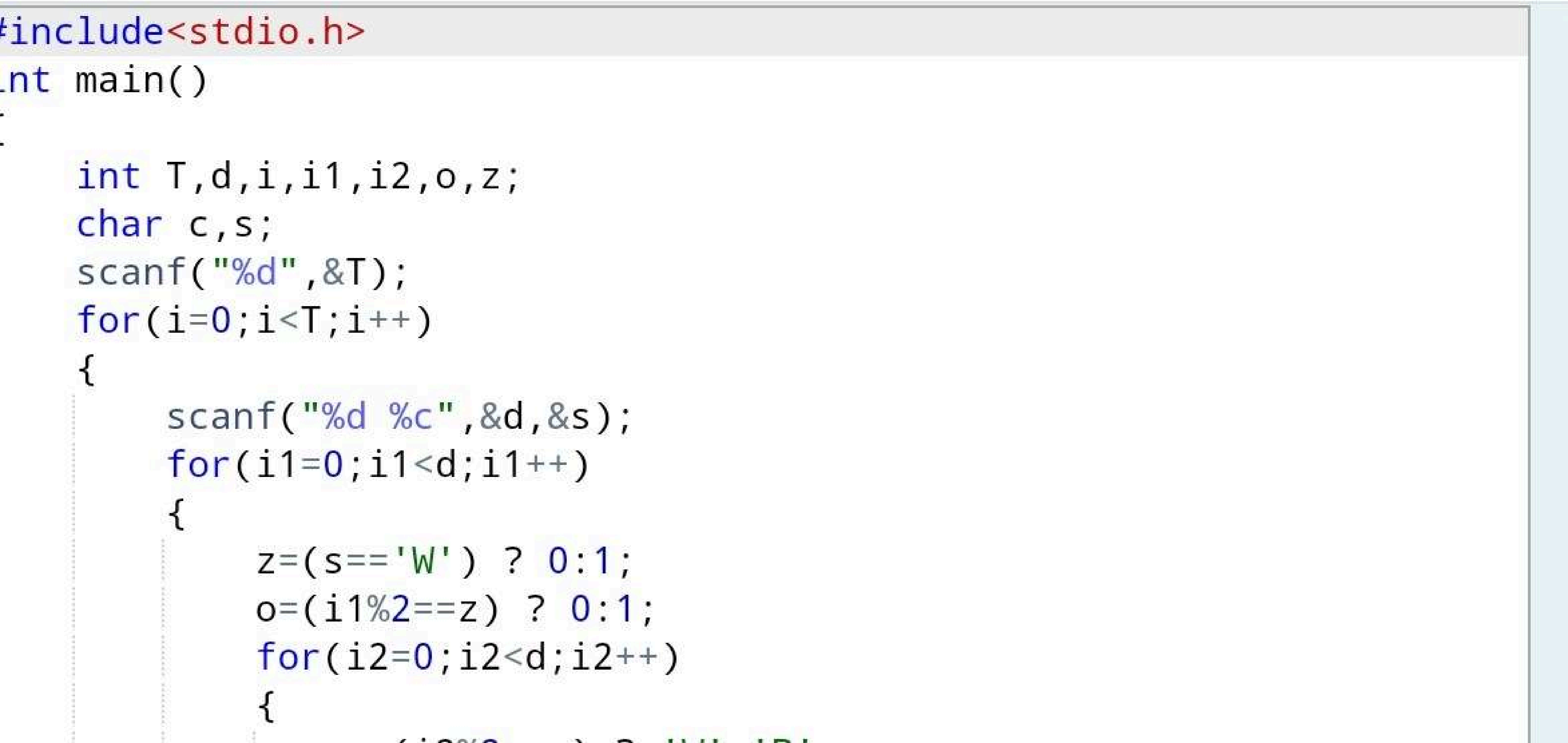
Write a program that takes input:

The first line contains T, the number of test cases

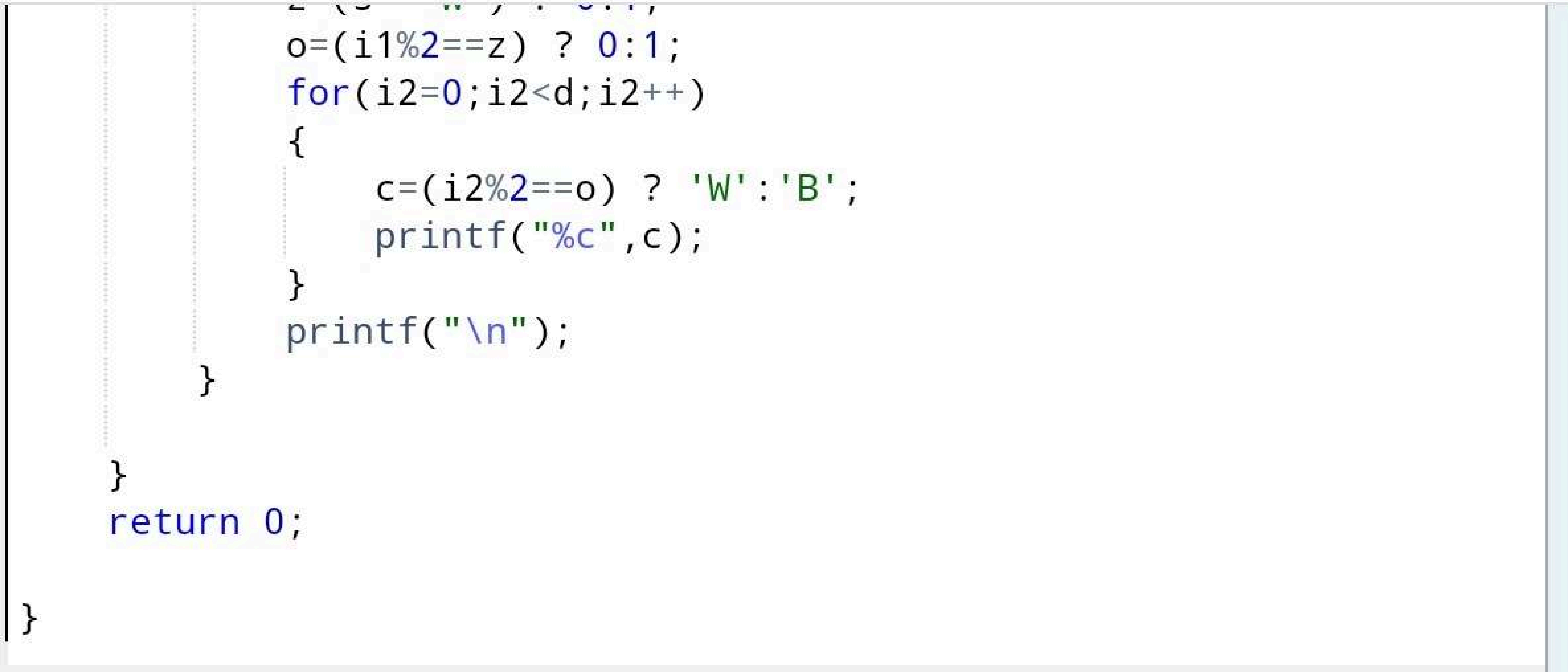
Each test case contains an integer N and also the starting character of the chessboard

Output Format

Print the chessboard as per the given examples



```
#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++)
            {
```



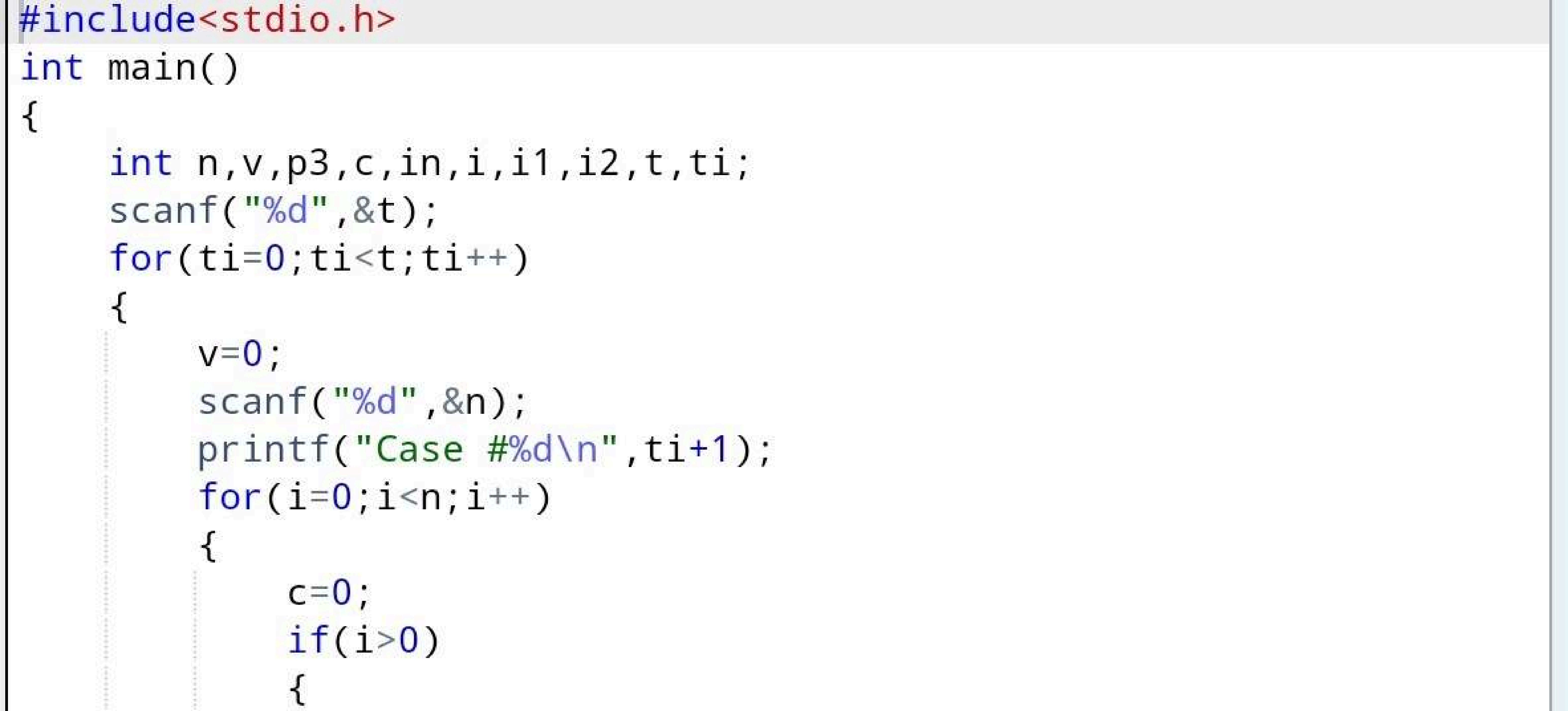
```
            o=(i1%2==z) ? 0:1;
            for(i2=0;i2<d;i2++)
            {
                c=(i2%2==o) ? 'W':'B';
                printf("%c",c);
            }
            printf("\n");
        }

    }
    return 0;

}
```

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

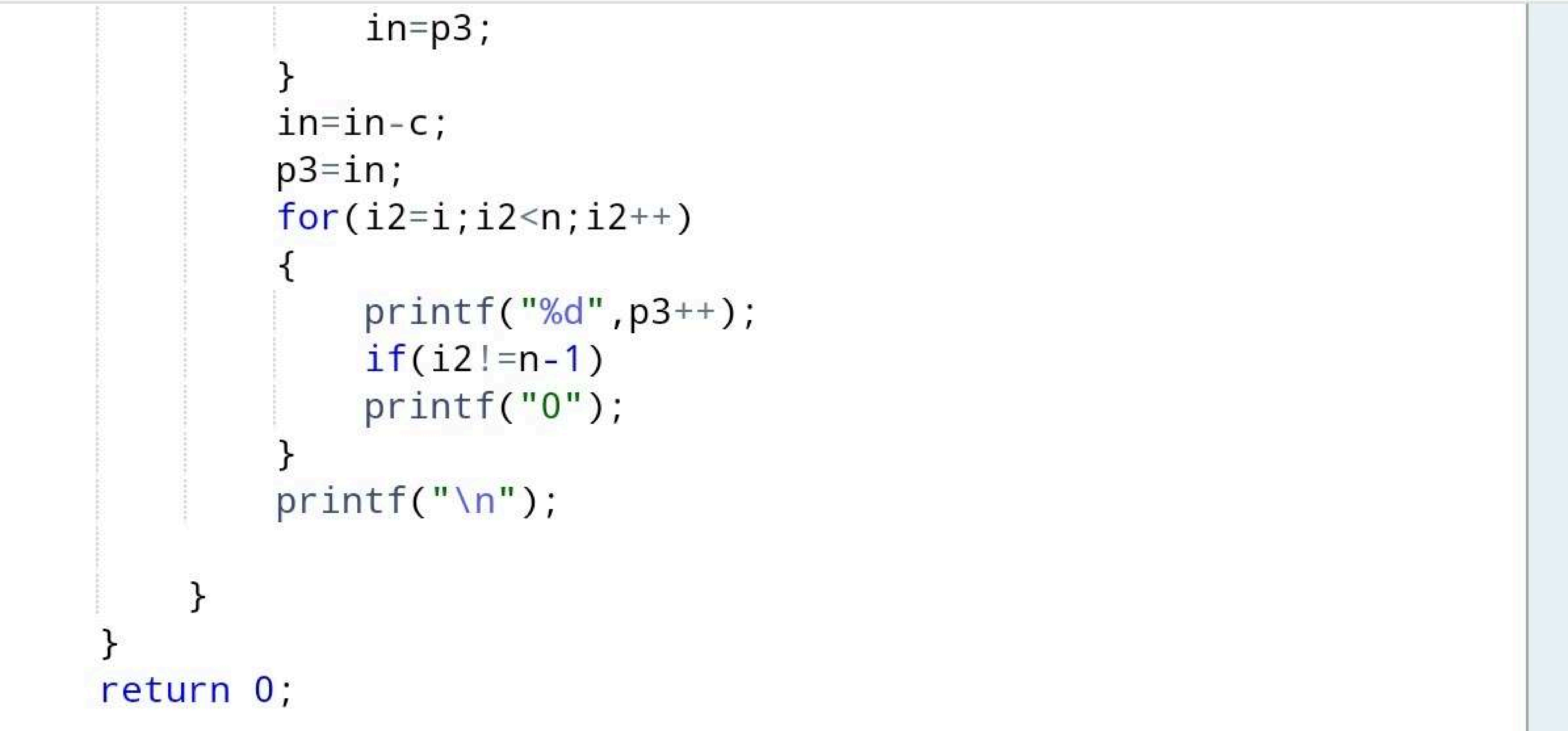
```
10203010011012

**4050809

****607
```

If N= 4, then pattern will be:

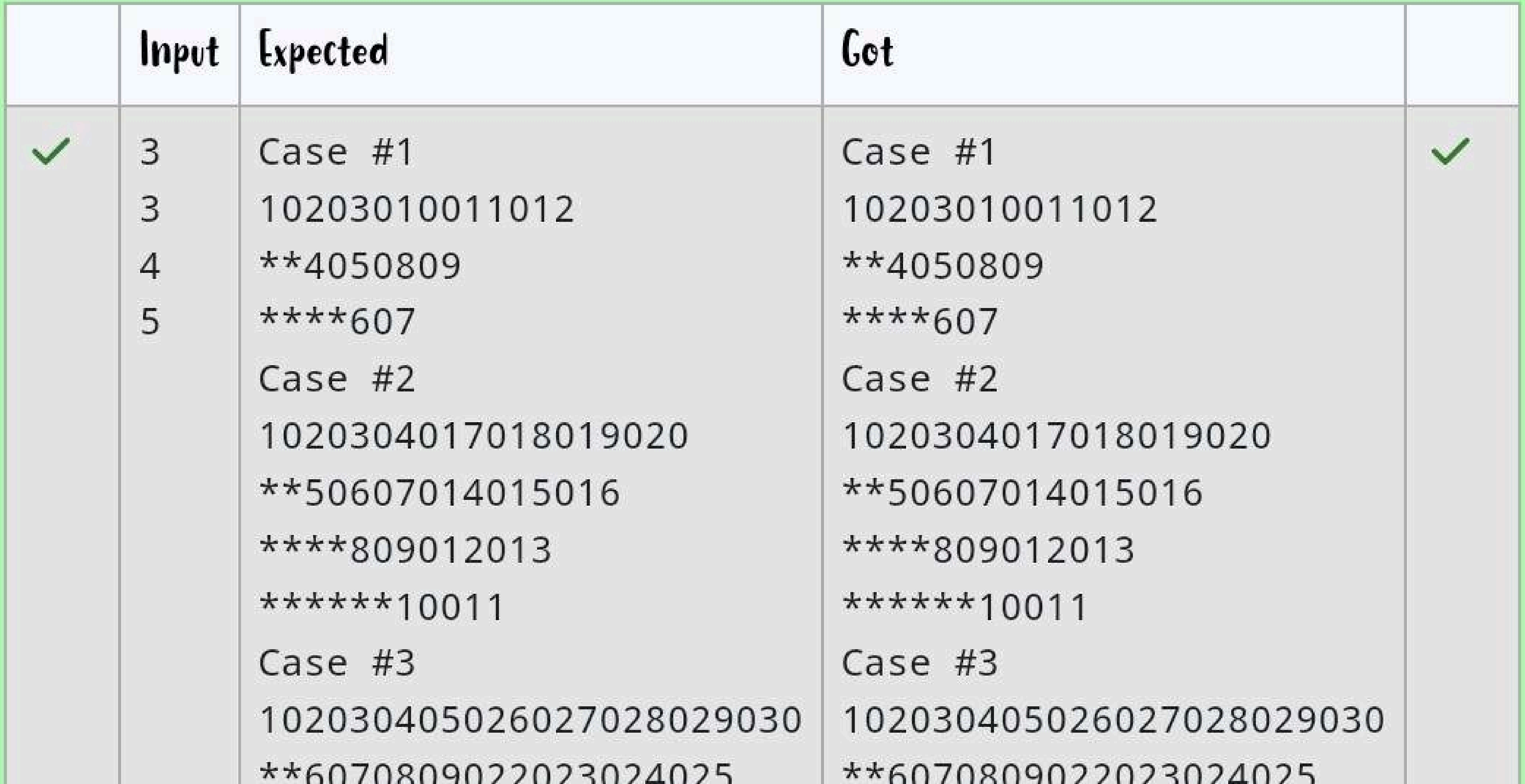
```
10203040170180190 20

**50607014015016

****809012013
```



```
#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)
            {
```



```
                in=p3;
            }
            in=in-c;
            p3=in;
            for(i2=i;i2<n;i2++)
            {
                printf("%d",p3++);
                if(i2!=n-1)
                printf("0");
            }
            printf("\n");

        }
    }
    return 0;
```



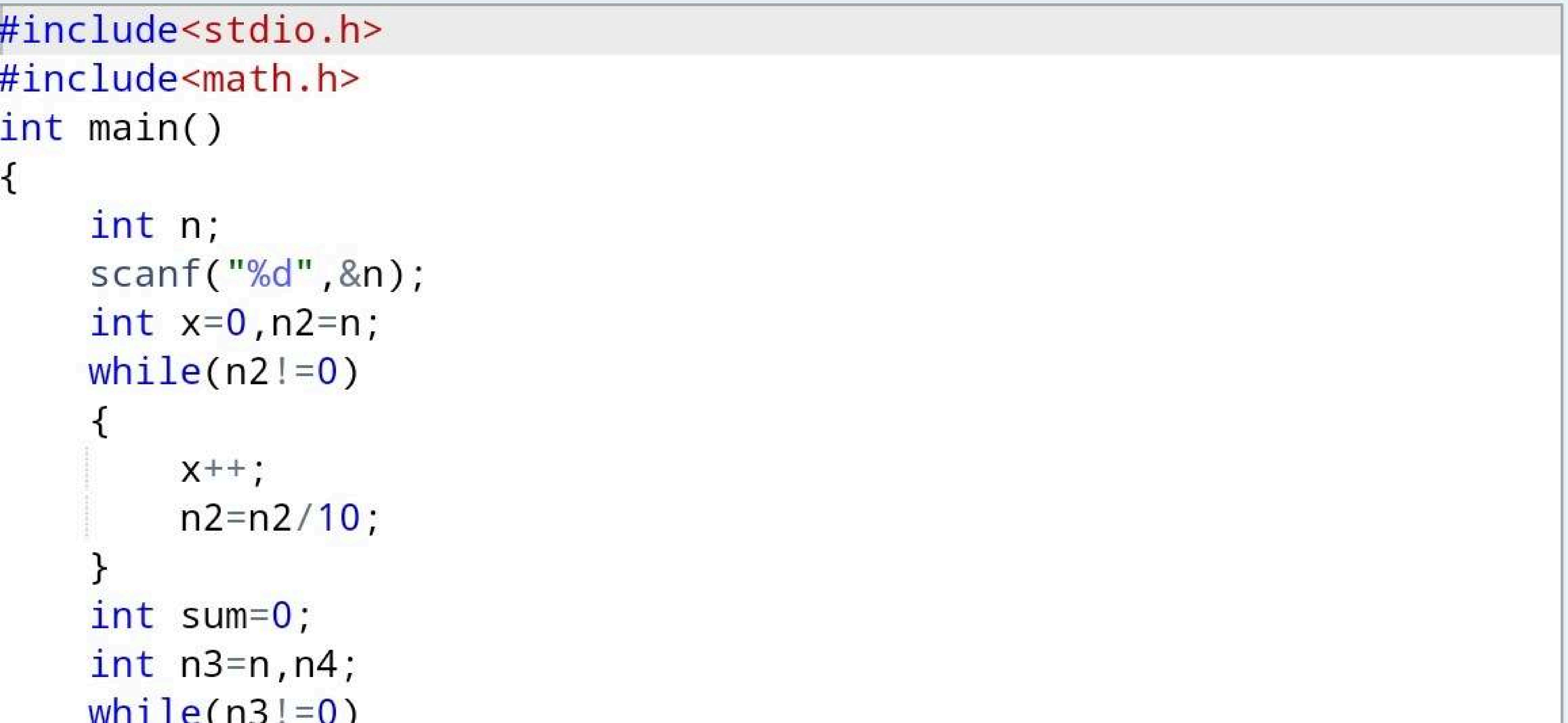
	Input	Expected	Got	
✓	3 3 4 5	Case #1 10203010011012 **4050809 ****607 Case #2 102030401701801 9020 **50607014015016 ****809012013 ******10011 Case #3 102030405026027028029030 **6070809022023024025	Case #1 10203010011012 **4050809 ****607 Case #2 102030401701801 9020 **50607014015016 ****809012013 ******10011 Case #3 102030405026027028029030 **6070809022023024025	✓

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

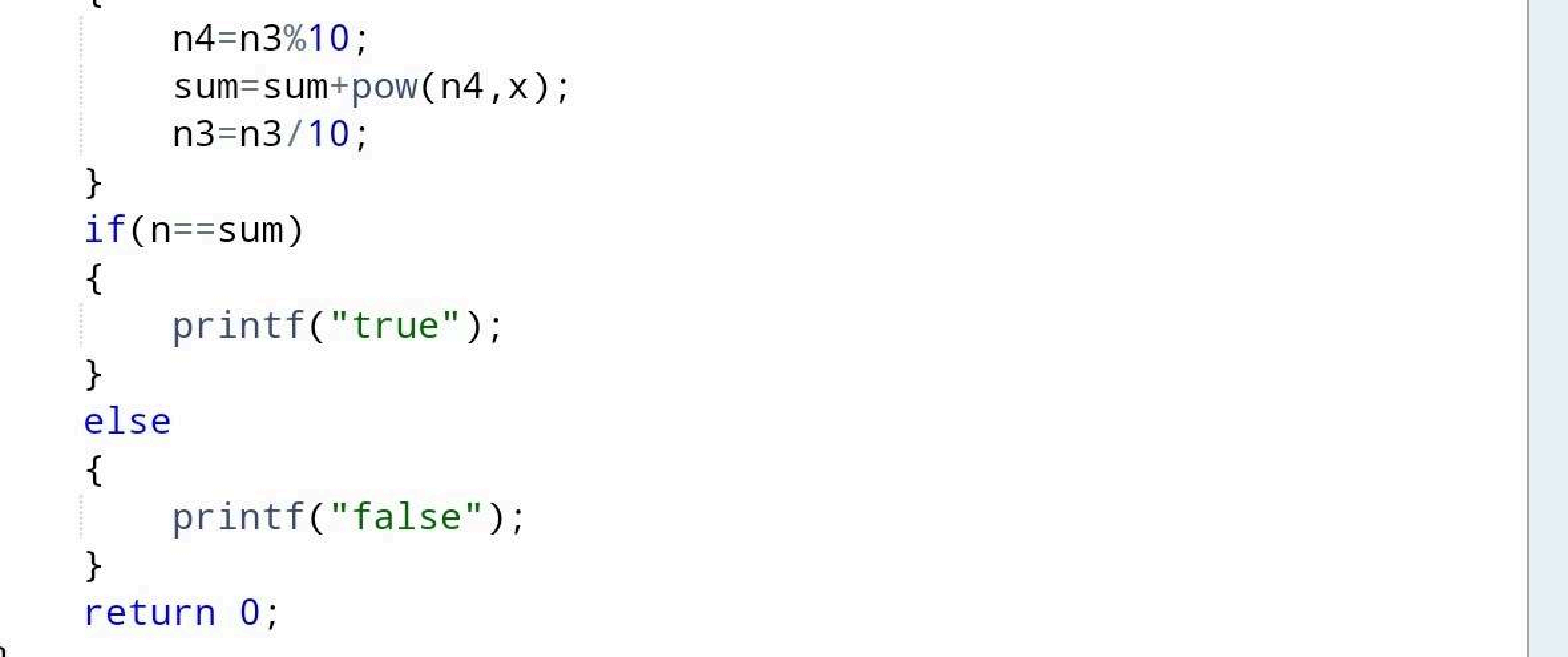
Given a positive integer N, return true if and only if it is an Armstrong number.

Example 1:

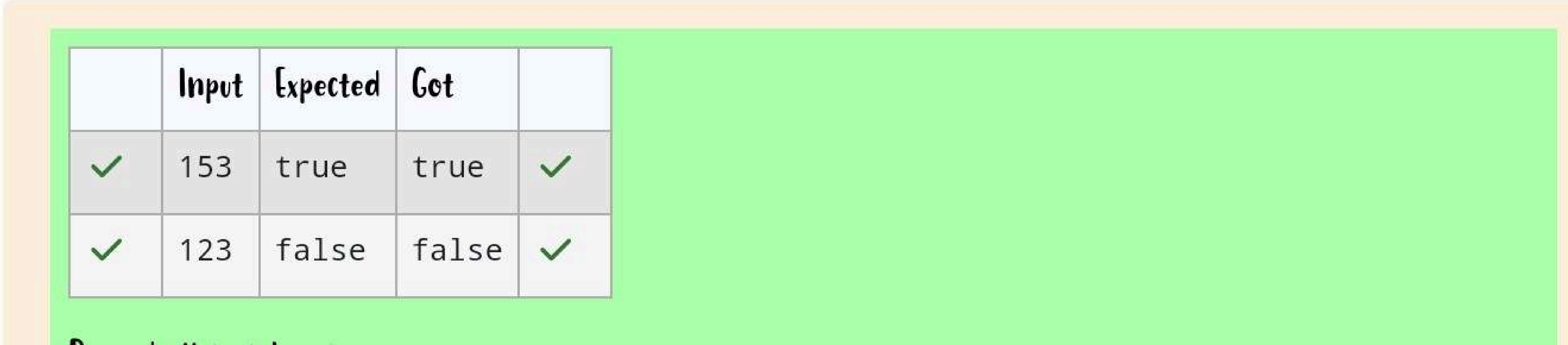
Input:



```
#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)
```



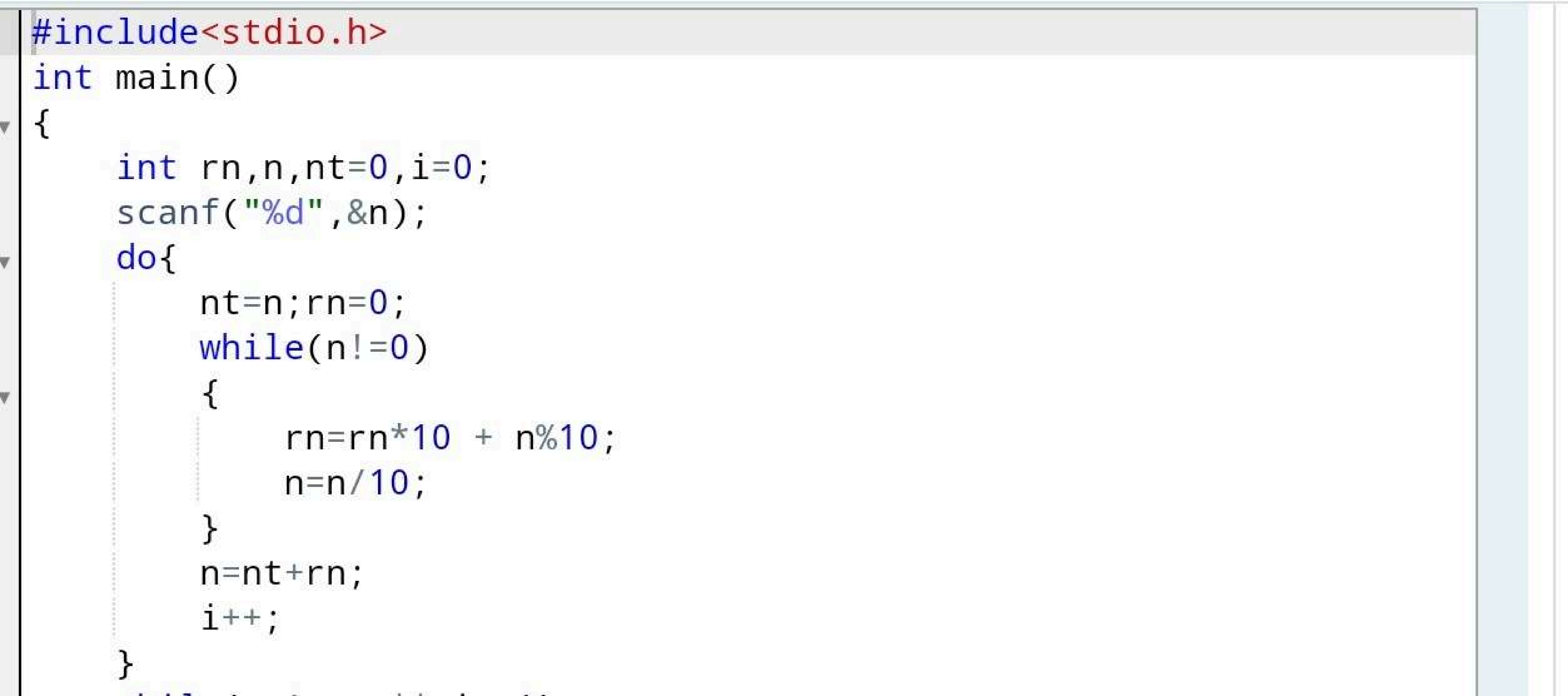
```
    {
        n4=n3%10;
        sum=sum+pow(n4,x);
        n3=n3/10;
    }
    if(n==sum)
    {
        printf("true");
    }
    else
    {
        printf("false");
    }
    return 0;
}
```



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

Take a number, reverse it and add it to the original number until the obtained number is a palindrome.
Constraints 1<=num<=99999999 Sample Input 1 32 Sample Output 1 55 Sample Input 2 789 Sample Output 2 66066

Answer: (penalty regime: 0 %)



```c
#include<stdio.h>
int main()
{
    int rn,n,nt=0,i=0;
    scanf("%d",&n);
    do{
        nt=n;rn=0;
        while(n!=0)
        {
            rn=rn*10 + n%10;
            n=n/10;
        }
        n=nt+rn;
        i++;
    }
```

```
        while(n>0)
        {
            rn=rn*10 + n%10;
            n=n/10;
        }
        n=nt+rn;
        i++;
    }
    while(rn!=nt || i==1);
    printf("%d",rn);
    return 0;
}
```

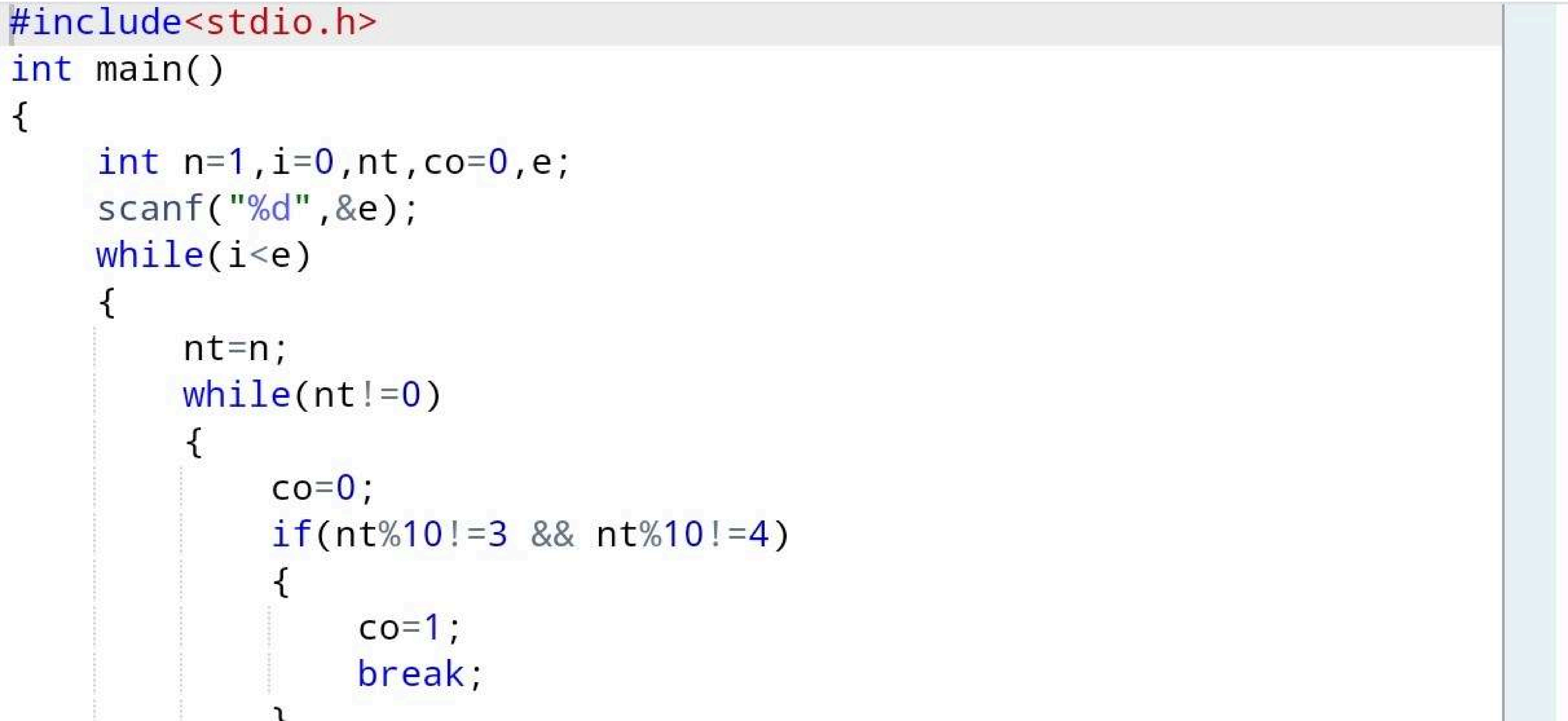
	Input	Expected	Got	
✓	32	55	55	✓
✓	789	66066	66066	✓

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.

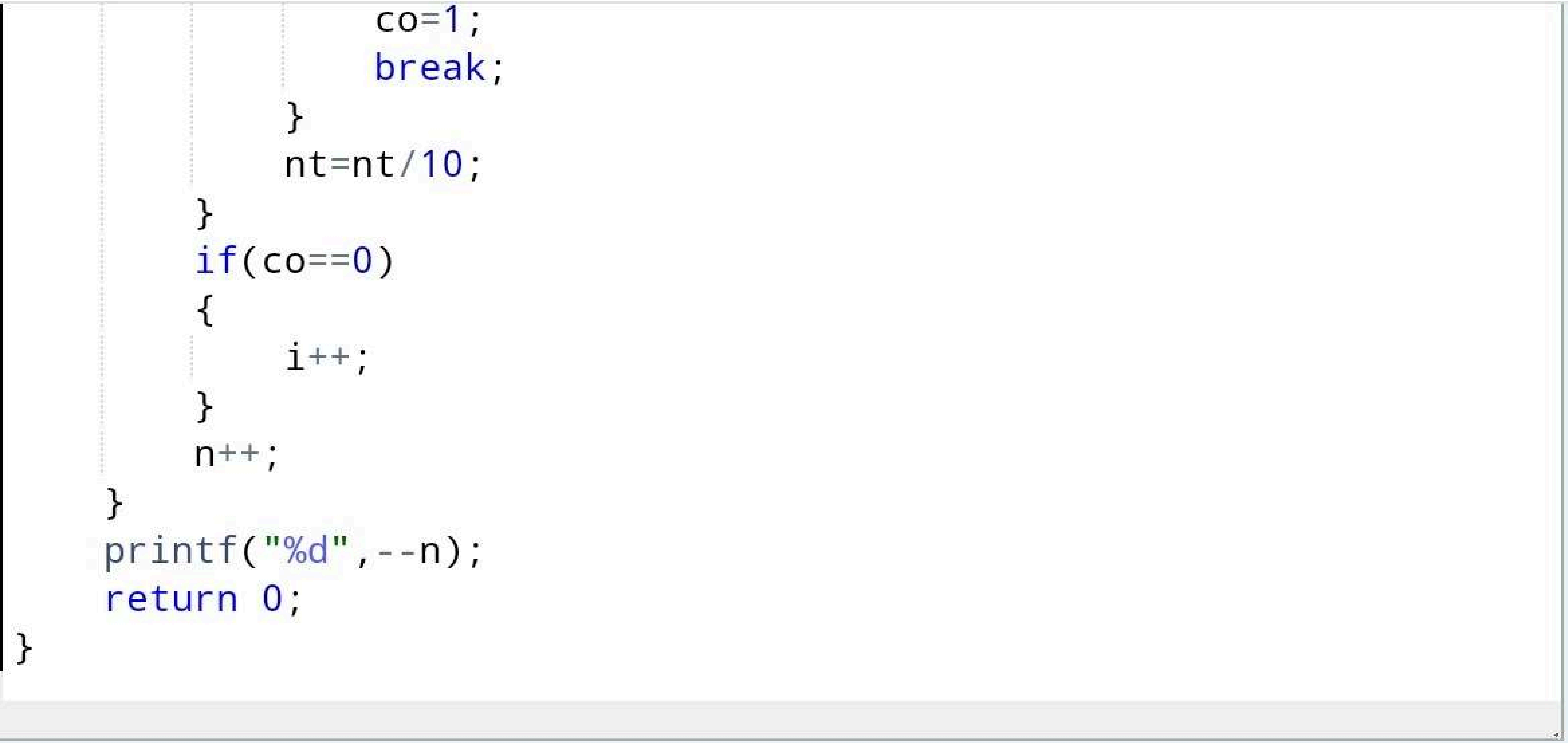
The program should accept a number 'n' as input and display the nth lucky number as output.

Sample Input 1:

```
3
```



```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;
            }
```



```
                co=1;
                break;
            }
            nt=nt/10;
        }
        if(co==0)
        {
            i++;
        }
        n++;
    }
    printf("%d",--n);
    return 0;
}
```

	Input	Expected	Got	
✓	34	33344	33344	✓