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

```
Input Expected Got
                       WBW
            BWB
                       BWB
            WBW
                      WBW
            WBWBW
                      WBWBW
            BWBWB
            WBWBW
                      WBWBW
            BWBWB
                      BWBWB
            WBWBW
                      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

Question $\bf 3$ Decode the logic and print the Pattern that corresponds to given input. Correct Marked out of 7.00 If N= 3 Flag question 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 First line print Case #i where i is the test case number In the subsequent line, print the pattern Test Case 1 3 3 4 5 Output Case #1 10203010011012 **4050809 ****607 Case #2 1020304017018019020 **50607014015016 ****809012013 *****10011 Case #3 102030405026027028029030 **6070809022023024025 ****10011012019020021 *****13014017018

******15016

```
int n,v,p3,c,in,i,i1,i2,t,ti;
scanf("%d",&t);
for(ti=0;ti<t;ti++)</pre>
                     v=0;
scanf("%d",&n);
printf("Case #%d\n",ti+1);
for(i=0;i<n;i++)</pre>
10
11
12 v
13
14
15 v
16
17 v
18
19
20
21
22 v
23
24 v
25
26
27
28
29
30 v
31
32
33
34
35
36
37 v
38
39
40 v
41
42
43
44
44
45
46
47 }
                          { 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)
                            if(i2!=n-1)
{
    printf("0");
}
}printf("\n");
               }
return 0;
```

	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	

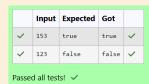
Question 1
Correct
Marked out of 3.00
Flag question

Answer: (penalty regime: 0 %)

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 Explanation: 123 is a 3-digit number, and 123 != 1^3 + 2^3 + 3^3 = 36. Example 3: Input: 1634 Output: true Note: 1 <= N <= 10^8

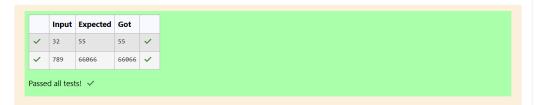
```
#include <stdio.h>
#include <math.h>
int main()
     4
5
6
7
                  int num,n,a=0,d=0,b=0,s;
scanf("%d",&num);
n=num;
                 b=num;
while(n!=0)
   10
11
                        n=n/10;
12
13
14
15 *
16
17 *
18
19
20
21
22
23
24
25
26
27
28 }
                        d++;
                 } while(b!=0)
                 {
                       for(int i=1;i<=d;i++)
                      s=b%10;
b=b/10;
a=a+(pow(s,d));
}
                              s=b%10;
                       if(num==a)
printf("true");
                        else
printf("false");
                  return 0;
```



Question 2 Correct Marked out of ▼ Flag question

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

```
int n,nt=0,t=0,rn;
scanf("%d",&n);
do
                  rn=0;
while(n!=0)
                 {
    rn=rn*10+n%10;
    n=n/10;
}
 11 v
 12
13
14
15
                   n=nt+rn;
             t++;
}while(rn!=nt || t==1);
printf("%d",rn);
return 0;
 16
17
18
19
 20 }
```



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:

2

Sample Output 1:

33

Explanation:

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

Sample Input 2:

2/

Sample Output 2:

33344

```
Answer: (penalty regime: 0 %)
```

