POOJA SREE P

ECE-D

```
240801241 ,
```

Problem Statement:

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 value for size of the chessboard

Output format:

Print a chessboard of dimensions size * size.

Print W for white spaces and B for black spaces.

Sample Input:

2

3

5

Sample Output:

WBW

BWB

WBW

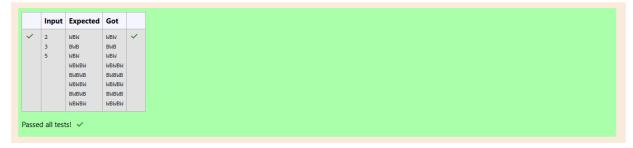
WBWBW

BWBWB

WBWBW

BWBWB

WBWBW



Problem Statement:

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:

2

2 W

3 B

Sample Output:

WB

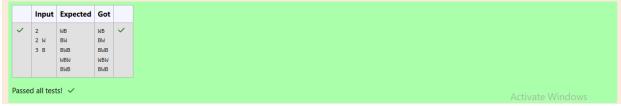
BW

BWB

WBW

BWB





Problem Statement:

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 ${\sf N}$

```
Output Format
First line print Case #i where i is the test case number, In the subsequent line,
print the
pattern
Sample Input
3
3
4
5
Sample Output
Case #1
10203010011012
**4050809
****607
Case #2
1020304017018019020
**50607014015016
****809012013
*****10011
Case #3
102030405026027028029030
**6070809022023024025
****10011012019020021
******13014017018
*******15016
```

```
1  ##nclude <stdio.h>
2
3  int main(){
4    int t;
5    scanf("Md", %t);
6    for(int x=1;x<=t;x++){
7    printf("dd", %n);
int n;
8    int n;
9    scanf("Md", %n);
int f=1,b=n*(n+1);
10    for(int i=0;icn;z++){
11    for(int i=0;icn;z++){
12    printf("d", f);
13    printf("d", f);
14    printf("d", f);
15    for(int b=0;cn;z++){
16    printf("d", f);
17    for(int b=0;cn;z++){
19    for(int b=0;cn;z++){
19    printf("d", f);
19    f++;
20    }
21    for(int l=b-(n-i)+1;l<-b;l++){
22    printf("d", f);
33    }
44    b=n-i;
25    printf("d", f);
34    printf("d", f);
35    printf("d", f);
36    printf("d", f);
37    printf("d", f);
38    printf("d", f);
39    printf("d", f);
30    printf("d", f);
31    printf("d", f);
32    printf("d", f);
33    printf("d", f);
34    printf("d", f);
35    printf("d", f);
36    printf("d", f);
37    printf("d", f);
38    printf("d", f);
39    printf("d", f);
30    printf("d", f);
31    printf("d", f);
32    printf("d", f);
33    printf("d", f);
34    printf("d", f);
35    printf("d", f);
36    printf("d", f);
37    printf("d", f);
38    printf("d", f);
39    printf("d", f);
30    printf("d", f);
31    printf("d", f);
32    printf("d", f);
33    printf("d", f);
34    printf("d", f);
35    printf("d", f);
36    printf("d", f);
37    printf("d", f);
38    printf("d", f);
39    printf("d", f);
30    printf("d", f);
31    printf("d", f);
32    printf("d", f);
33    printf("d", f);
34    printf("d", f);
35    printf("d", f);
36    printf("d", f);
37    printf("d", f);
38    printf("d", f);
39    printf("d", f);
30    printf("d", f);
30    printf("d", f);
31    printf("d", f);
31    printf("d", f);
32    printf("d", f);
33    printf("d", f);
34    printf("d", f);
35    printf("d", f);
36    printf("d", f);
37    printf("d", f);
38    printf("d", f);
39    printf("d", f);
30    printf("d", f);
30    printf("d", f);
31    printf("d", f);
32    printf("d", f);
33    printf("d", f);
34    printf("d", f);
35    printf("d", f);
36    printf("d", f);
37    printf("d", f);
38    printf("d
```



Problem Statement:

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.

Note: 1 <= N <= 10^8

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

Sample Input:

153

Sample Output:

true

Sample Input:

123

Sample Output:

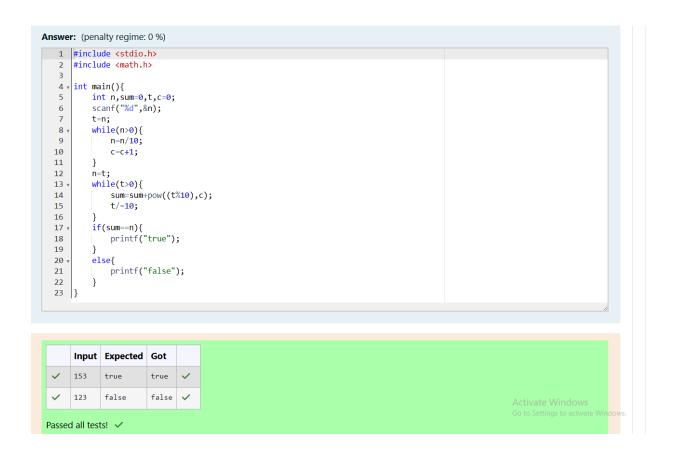
false

Sample Input:

1634

Sample Output:

true



Problem Statement:

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 %)
   1 #include <stdio.h>
   4 v int rev(int n){
          int reverse=0;
          while(n>0){
          reverse=(reverse*10)+(n%10);
          n=n/10;
  10
          return reverse;
  11 }
  12 v int Pal(int n){
          return rev(n)==n;
  13
  14
  15 v int main(){
          int n;
scanf("%d",&n);
  16
  17
          while(!Pal(n)){
  18 🔻
  19
             int r=rev(n);
  20
             n=n+r;
  21
          printf("%d",n);
  22
  23 }
```

Problem Statement:

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

```
Answer: (penalty regime: 0 %)
  1 #include <stdio.h>
    3 1
        int main(){
    4
5
            int n=1;
           int i=0,ns,c=0,e;
scanf("%d",&e);
while(i<e){</pre>
    6
    7 •
                ns=n;
while(ns!=0){
    8
   9 ,
                    c=0;
if(ns%10!=3&&ns%10!=4){
   10
  11 1
                     c=1;
break;
  12
  13
  14
                     ns/=10;
  15
  16
                 if(c==0){
  17
                i++;
  18
  19
  20
                 n++;
   21
            printf("%d",--n);
   22
```

