WEEK 5

Simple Chessboard

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

```c
#include<stdio.h>
int main()
{
    int T, size;
    scanf("%d", &T);
    for(int t = 0; t< T; t++)
    {
        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! ✓

Print Our Own Chessboard

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

```c
#include<stdio.h>
int main()
{
    int T,N;
    char ch;
    scanf("%d", &T);
    while(T--)
    {
        scanf("%d %c", &N, &ch);
        for (int i =0; i < N; i++)
        {
            for (int j = 0; j < N; j++)
            {
                if(ch == 'W')
                {
                    if((i + j) %2 == 0)
                    printf("W");
                    else
                    printf("B");
                }
                else
                {
                    if((i+j)%2 == 0)
                    printf("B");
                    else
                    printf("W");
                }
            }
            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! ✓

Pattern Printing

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:

10203040170180190 20

**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 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

10203040502602702802903O

**6070809022023024025

****10011012019020021

******13014017018

```c
#include<stdio.h>
int main()
{
    int t,n,x,y,z=1,i,ans,c;
    scanf("%d", &t);
    while(z<=t)
    {
        scanf("%d", &n);
        printf("Case #%d\n",z);
        y=1;
        i=1;
        c=0;
        while(y<=n)
        {
            x=1;
            ans = (n*n);
            ans = ans-c;
            while(x<=2*n)
            {
                if(x<=n)
                {
                    if(x<y)
                    printf("**");
                    else if (x<=n)
                    {
                        printf("%d",i*10);
                        i++;
                    }
                }
                else
                {
                    if((x+y)==(2*n)+1)
                    {
```

```
                    ans++;
                    c++;
                }
                else if (x+y<=(2*n)+1)
                {
                    printf("%d",(ans+y)*10);
                    ans++;
                    c++;
                }
            }
            x++;
        }
        y++;
        printf("\n");
    }
    z++;
}
return 0;
```

	Input	Expected	Got	
✓	3 3 4 5	Case #1 10203010011012 **4050809 ****607 Case #2 1020304017018019020 **50607014015016 ****809012013 ******10011 Case #3 10203040502602702802903 0 **6070809022023024025 ****10011012019020021 ******13014017018 ********15016	Case #1 10203010011012 **4050809 ****607 Case #2 1020304017018019020 **50607014015016 ****809012013 ******10011 Case #3 10203040502602702802903 0 **6070809022023024025 ****10011012019020021 ******13014017018 ********15016	✓

Passed all tests! ✓

Armstrong Number

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

```c
#include<stdio.h>
#include<math.h>
int main()
{
    int N,k = 0 , sum =0 , rem;
    scanf("%d", &N);
    int temp1 = N, temp2 = N;
    while (temp1 != 0)
    {
        temp1/=10;
        k++;
    }
    while (temp2 != 0)
    {
        rem = temp2 % 10;
        sum+=pow(rem,k);
        temp2/=10;
    }
    if(sum== N)
    printf("true");
    else
    printf("false");
}
```

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

Passed all tests! ✓

Reverse and Add Until Get a Palindrome

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

```c
#include<stdio.h>
int main()
{
    long long int num, sum, revnum, tempnum, tempsum;
    scanf("%lld", &num);
    while(1)
    {
        revnum = 0;
        tempnum = num;
        while(num)
        {
            revnum = revnum*10 + (num%10);
            num = num/10;
        }
        sum = tempnum + revnum;
        tempsum = sum;
        revnum= 0;
        while(sum)
        {
            revnum = revnum*10+(sum%10);
            sum = sum/10;
        }
        if(tempsum == revnum)
        {
            break;
        }
        num = tempsum;
    }
    printf("%lld",tempsum);
    return 0;
}
```

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

Passed all tests! ✓

Lucky Number

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

```c
#include<stdio.h>
int main()
{
    long int i,j;
    int rem,n,count =0,flag;
    scanf("%d", &n);
    for(i=1; count<=n; i++)
    {
        flag = 0;
        j=i;
        while (j>0)
        {
            rem = j%10;
            if(rem == 3|| rem == 4)
            j = j/10;
            else
            {
                flag = 1;
                break;
            }
        }
        if(flag == 0)
        {
            count++;
            if(count == n)
            break;
        }
    }
    printf("%ld",i);
    return 0;
}
```

	Input	Expected	Got	
✓	34	33344	33344	✓

Passed all tests! ✓