Q1)

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

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

Passed all tests! ✓

Q2)

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

```c
#include <stdio.h>
int main(){
    int n,r,i,j;
    char c;
    scanf("%d", &n);
    while (n--){
        scanf("%d %c", &r, &c);
        for (i = 0; i<r; i++){
            for (j = 0; j<r; j++){
                if (c == '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! ✓

Q3)

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

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

Case #1

10203010011012

**4050809

****607

Case #2

1020304017018019020

**50607014015016

****809012013

******10011

Case #3

10203040502602702802 9030

**6070809022023024025

****10011012019020021

******13014017018

********15016

```c
#include <stdio.h>
int main(){
    int n,v,p3,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){
            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) printf("0");
        }printf("\n");
        }
    }
}
```

	Input	Expected	Got	
✓	3 3 4 5	Case #1 10203010011012 **4050809 ****607 Case #2 1020304017018019020 **50607014015016 ****809012013 ******10011 Case #3 10203040502602702802 9030 **6070809022023024025 ****10011012019020021 ******13014017018 ********15016	Case #1 10203010011012 **4050809 ****607 Case #2 1020304017018019020 **50607014015016 ****809012013 ******10011 Case #3 10203040502602702802 9030 **6070809022023024025 ****10011012019020021 ******13014017018 ********15016	✓

Passed all tests! ✓

Q4)

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

```c
#include <stdio.h>
#include <math.h>
int main(){
    int n;
    scanf("%d", &n);
    int n2 = n;
    int x = 0;
    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	✓

Passed all tests! ✓

Q5)

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

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

Passed all tests! ✓

Q6)

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

```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! ✓