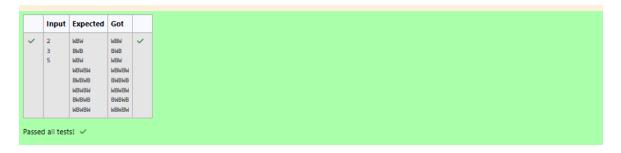
WEEK 5

| Q) 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 |
| |



Q) 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 W

3 B

Output:

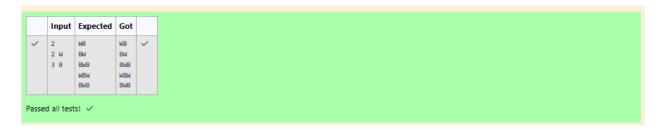
WB

BW

BWB

WBW

```
20
21
22 }
   return 0;
```



| Q) 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

Output

Case #1

10203010011012

**4050809

****607

Case #2

1020304017018019020

**50607014015016

****809012013

*****10011

Case #3

102030405026027028029030

**6070809022023024025

****10011012019020021

*****13014017018

******15016

```
(ti=0;it<;ti++){
v=0;
scanf("%d",&n);
printf("Case #%d\n",ti+1);
for(i=0;i<n;i++){</pre>
                                                                                                                                                                (10,20,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1) (10,1)
  10
11
    12
13
14
15
                                                                                                                                                           for(j=i;j<n;j++){
   if(i>0) C++;
   printf("%d0",++v);
16
17
18
19
                                                                                                                                                                if(i==0){
 p=v+(v*(v-1))+1;
 in=p;
  20
  22
                                                                                                                                                                   in=in-c;
  24
25
                                                                                                                                                                in=in-c;
p=in;
for(k=i;k<n;k++){
    printf("%d",p++);
    if(k!=n-1) printf("0");</pre>
  26
27
28
29
  30
31
                                                                                                                                                                } printf("\n");
    32
33
34 }
                                                                                       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 | |

Passed all tests! 🗸

| Q) 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

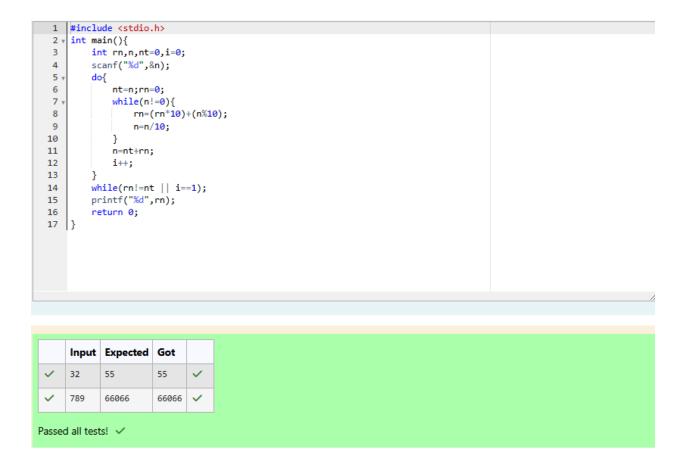
Note:

1 <= N <= 10^8

| | Input | Expected | Got | | |
|---------------------|-------|----------|-------|----------|--|
| ~ | 153 | true | true | ~ | |
| ~ | 123 | false | false | ~ | |
| Passed all tests! ✓ | | | | | |

Q) Take a number, reverse it and add it to the original number until the obtained number is a palindrome. Constraints 1<=num<=999999999 Sample Input 1 32 Sample Output1 55

Sample Input 2 789 Sample Output 2 66066



Q) 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 |
| |

```
1 #include <stdio.h>
  2 v int main(){
        int n=1,i=0,nt,co=0,e;
scanf("%d",&e);
  3
  4
         while(i<e){
  5 v
   6
             nt=n;
   7 ,
             while(nt!=0){
   8
                co=0;
  9 ,
                 if(nt%10!=3 && nt%10!=4){
                 co=1;
break;
  10
  11
                }
  12
               nt=nt/10;
  13
  14
              if(co==0){
  15 v
             `i++;
  16
  17
  18
             n++;
  19
          printf("%d",--n);
  20
          return 0;
  21
  22 }
```

| | Input | Expected | Got | |
|----------|-------|----------|-------|---|
| ~ | 34 | 33344 | 33344 | ~ |

Passed all tests! <