Ex. No.: 28 Date: 8.11.2024

# **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** 

 ${\sf WBWBW}$ 

**BWBWB** 

**WBWBW** 

#### **Program:** #include<stdio.h> int main() 2 3 ▼ { 4 int T,d,i=0,i1,i2; char c; scanf("%d",&T); 5 6 7 while(i<T)</pre> 8 scanf("%d",&d); 9 10 i1=0; 11 while(i1<d)</pre> 12 1 i2=0; 13 14 15 while(i2<d) 16 17 if((i1+i2)%2==0) 18 { c='W'; 19 20 21 else { c='B';} printf("%c",c); 22 23 i2++; 24 25 i1++; printf("\n"); 26 27 28 i++; 29 30 return 0; 31 } Input Expected Got 2 WBW BWB BWB 3 WBW WBW WBWBW WBWBW BWBWB BWBWB WBWBW WBWBW BWBWB BWBWB WBWBW WBWBW Passed all tests! <

Ex. No.: 29 Date: 8.11.2024

### **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

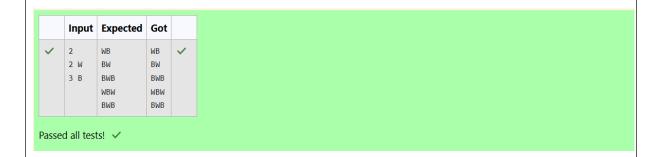
**BWB** 

WBW

**BWB** 

## **Program:**

```
#include<stdio.h>
    int main()
 2
 3 🔻
    {
 4
        int T,d,i,i1,i2,a,Z;
        char c,s;
scanf("%d",&T);
 5
 6
 7
        for(i=0;i<T;i++)</pre>
8
             scanf("%d %c",&d,&s);
9
             for(i1=0;i1<d;i1++)
10
11 1
                 Z=(s=='W')?0:1;
12
                 a=(i1%2==Z)?0:1;
13
14
                 for(i2=0;i2<d;i2++)
15 1
                     c=(i2%2==a)?'W':'B';
16
                     printf("%c",c);
17
18
                 printf("\n");
19
20
            }
21
        return 0;
22
23 }
```



Ex. No.: 30 Date: 8.11.2024

## **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: 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 Format**

First line print Case #i where i is the test case number, In the subsequent line, print the pattern

## Sample Input

3

4 5

Sample Output

Case #1

10203010011012

\*\*4050809

\*\*\*\*607

Case #2

1020304017018019020

\*\*50607014015016

\*\*\*\*809012013

\*\*\*\*\*10011

Case #3

102030405026027028029030

\*\*6070809022023024025

\*\*\*\*10011012019020021

\*\*\*\*\*13014017018

\*\*\*\*\*\*15016

```
Program:
       #include<stdio.h>
   2
       int main()
   3
       {
           int n,v,p,c,in,i,i1,i2,t,ti;
scanf("%d",&t);
   4
   5
   6
           for(ti=0;ti<t;ti++)</pre>
   7
   8
                v=0;
                scanf("%d",&n);
   9
                printf("Case #%d\n",ti+1);
  10
                for(i=0;i<n;i++)</pre>
  11
  12
  13
                    if(i>0)
  14
  15
  16
                        for(i1=0;i1<i;i1++)</pre>
                            printf("**");
  17
  18
                        for(i1=i;i1<n;i1++)</pre>
  19
  20
                             if(i>0)
  21
                            c++;
printf("%d0",++v);
  22
  23
  24
                        if(i==0)
  25
  26
  27
                             p=v+(v*(v-1))+1;
  28
                             in=p;
  29
  30
                    in=in-c:
                   p=in;
  31
  32
                    for(i2=i;i2<n;i2++)</pre>
  33
                        printf("%d",p++);
  34
  35
                        if(i2!=n-1)
                   printf("0");
}printf("\n");
  36
  37
  38
  39
  40
          return 0;
  41
  42
                                        Got
       Input Expected
       3
              Case #1
                                        Case #1
       3
              10203010011012
                                        10203010011012
       4
              **4050809
                                         **4050809
              ****607
                                        ****607
              Case #2
                                        Case #2
              1020304017018019020
                                        1020304017018019020
              **50607014015016
                                        **50607014015016
              ****809012013
                                        ****809012013
              *****10011
                                        *****10011
              102030405026027028029030 102030405026027028029030
              **6070809022023024025
                                         **6070809022023024025
              ****10011012019020021
                                       ****10011012019020021
              *****13014017018
                                        *****13014017018
              ******15016
                                        *******15016
 Passed all tests! <
```

Ex. No.: 31 Date: 13.11.2024

## **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

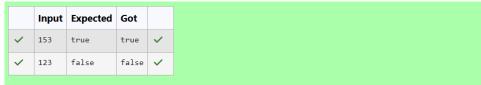
Sample **Input**:

1634

Sample **Output**:

true

```
Program:
        #include<stdio.h>
        #include<math.h>
        int main(){
       int n,k,r,s=0,d=0;
scanf("%d",&n);
     5
       k=n;
    7
        while(k!=0)
    8 ₹ {
    9
             k=k/10;
            d++;
   10
   11
   12
       k=n;
   13 while(k!=0)
14 v {
            r=k%10;
   15
   16
             s=s+pow(r,d);
   17
             k=k/10;
   18
        if(s==n) printf("true");
   19
        else printf("false");
return 0;
   20 el
21 re
22 }
```



Passed all tests! <

Ex. No.: 32 Date: 13.11.2024

## 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

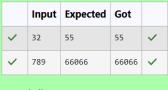
Sample Output 1

Sample Input 2 789

Sample Output 2 66066

## **Program:**

```
1 #include<stdio.h>
 2 int main()
 3 ▼ {
        int rn,n,nt=0,i=0;
scanf("%d",&n);
 4
 5
 6 ,
        do{
 7
            nt=n;rn=0;
 8
            while(n!=0)
 9 ,
10
                 rn=rn*10+n%10;
11
                n=n/10;
12
13
            n=nt+rn;
14
            i++;
15
16
        while(rn!=nt||i==1);
        printf("%d",rn);
17
18
         return 0;
19 }
```



Passed all tests! <

Ex. No.: 33 Date: 13.11.2024

## **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

```
Program:
   1 #include<stdio.h>
      int main()
   3 ₹
   4 int n=1,i=0,nt,co=0,e;
   5
      scanf("%d",&e);
   6
      while(i<e)
   8
          nt=n;
          while(nt!=0)
   9
  10
  11
              co=0;
              if(nt%10!=3 && nt%10!=4)
  12
  13 1
                 co=1;
  14
                 break;
  15
  16
              nt=nt/10;
  17
  18
          if(co==0)
  19
  20
  21
              i++;
  22
  23
  24
  25 printf("%d",--n);
  26
      return 0;
  27
  28 }
      Input Expected Got
            33344
                      33344 🗸
     34
Passed all tests! <
```