Day-2

Check for Symmetry

-62:18:11 Edit

Description

You are given a square matrix of size n. Rows are indexed 1 to n from top to bottom and columns are indexed 1 to n form left to right. Matrix consists of only '*' and '.'. You need t o check whether matrix is symmetric or not. if it is, check it is symmetric about vertical axis or horizontal axis or both.

A matrix is said to be symmetric about horizontal axis if 1st row is identical to n-th row, 2nd is identical to (n-1)th row and so on.

A matrix is said to be symmetric about vertical axis if 1ST column is identical to nth column, 2nd identical to (n-1)th and so on for all columns.

Input

Input Format :

First line contains t,the number of test cases. First line of each test case contains n the size of matrix. Each of next n lines contain n characters.

Constraints :

1<=t<=500

1<n<50

Output

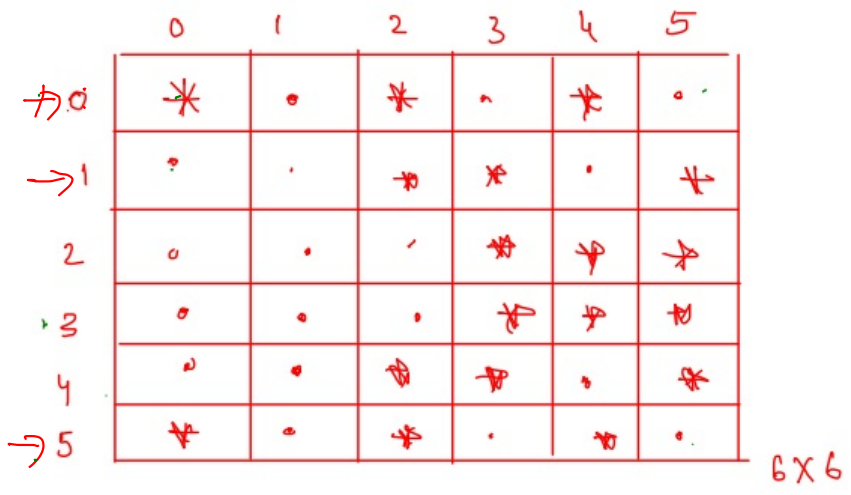
Output t lines, answer for each test case. Print "HORIZONTAL" if symmetric about horizontal axis. Print "VERTICAL" if symmetric about vertical axis. Print "BOTH" if symmetric ab out both axes. print "NO" if it is not symmetric.

Sample Input 1

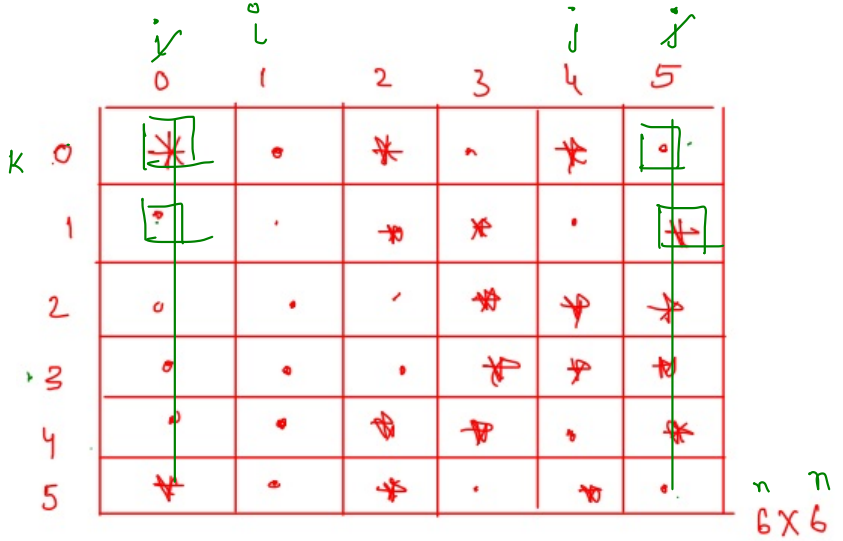
```
3
4
*.*.
.*.*
*.*.
.*.*
3
.*.
*.*
.*.
3
..*
**.
..*
```

Sample Output 1

```
NO
BOTH
HORIZONTAL
```




```
function hortizontal(arr[][],n)
{
        i=0,j=n-1,k=0
        while(i<j)
        {
                for(k=0;k<n;k++)
                {
                        if(arr[i][k]!=arr[j][k])
                                return false
                }
                i++
                j--
        }
        return true
}
```




```
main()
{
        h=horizontal(arr,n)
        v=vertical(arr,n)
        if(h==true && v==true)
        {
                print("Both")
        }
        else if(h==true && v==false)
        {
                print("horizontal")
        }
        else if(h==false && v==true)
        {
                print("vertical")
        }
        else
                print("no")
}
```

n^2

n^2

O(1)

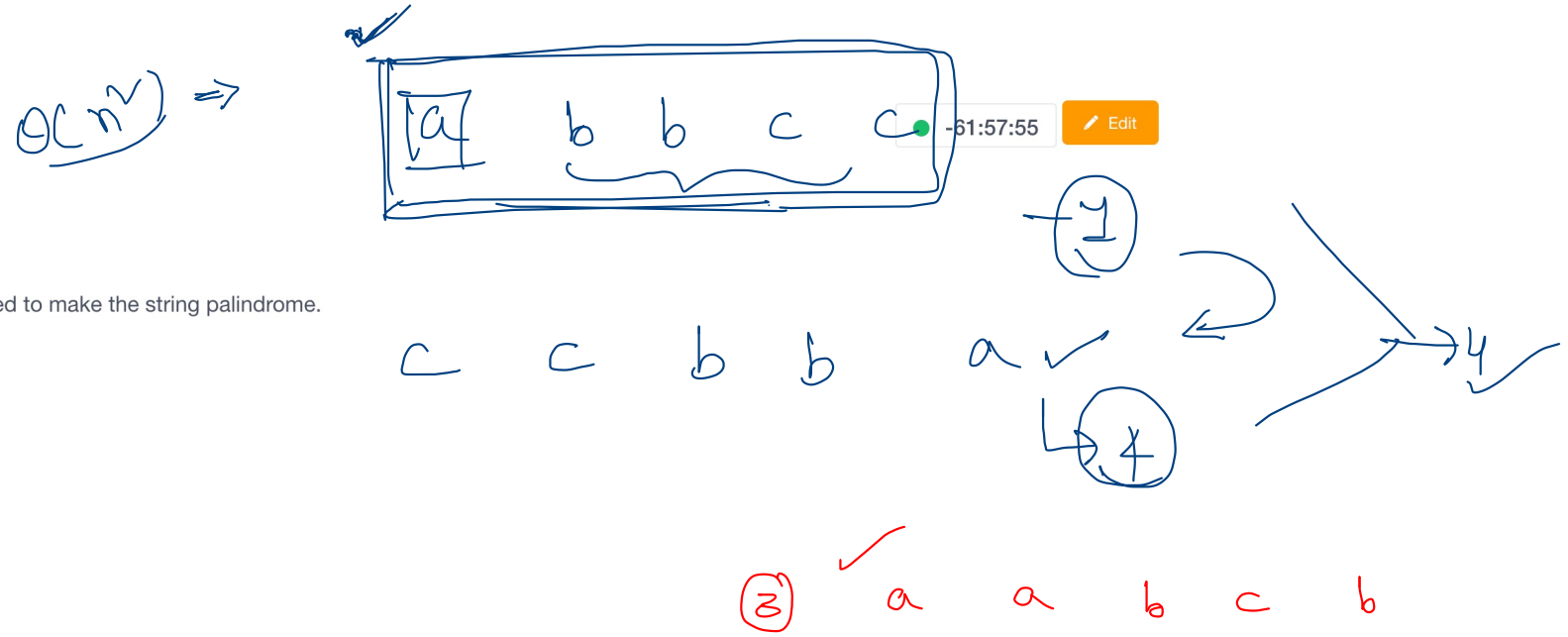
O(n^2) T.C

O(1) S.C

```
function vertical(arr[][],n)
{
        i=0,j=n-1,k=0
        while(i<j)
        {
                for(k=0;k<n;k++)
                {
                        if(arr[k][i]!=arr[k][j])
                                return false
                }
                i++
                j--
        }
        return true
}
```

⇒ Calls

Minimum Swaps to Make Palindrome



Description

You are given a string A, of length N.

You have to find the minimum number of adjacent swaps required to make the string palindrome.

If it is impossible, return -1.

Input

The first line of the input contains T, the number of test cases.

The next line of the input contains N, the length of the string.

The next line contains the string A itself.

Constraints

1 <= T <= 10

1 <= N <= 1000

Output

For each test case, print a single integer denoting the number of adjacent swaps required to make the string palindrome, on a new line.

Sample Input 1

```
2
5
aabcb
8
adbcdbad
```

Sample Output 1

```
3
-1
```

Hint

In the first sample test case, the given string is A = "aabcb".

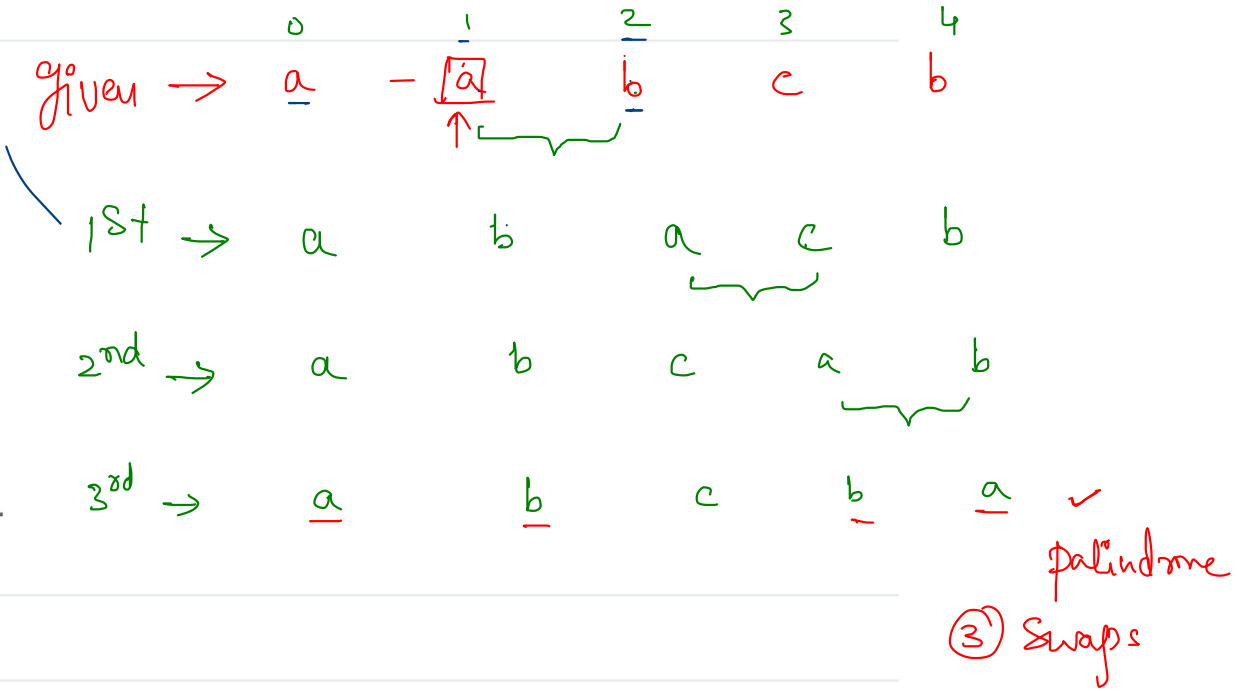
After the first swap, the string becomes -> "abacb".

After the second swap, the string becomes -> "abcab"

After the third swap, the string becomes -> "abcba", which is a palindrome.

So, in 3 adjacent swaps, the string became a palindrome, so the required output is 3.

In the second sample test case, the string cannot be made a palindrome, so the output is -1.

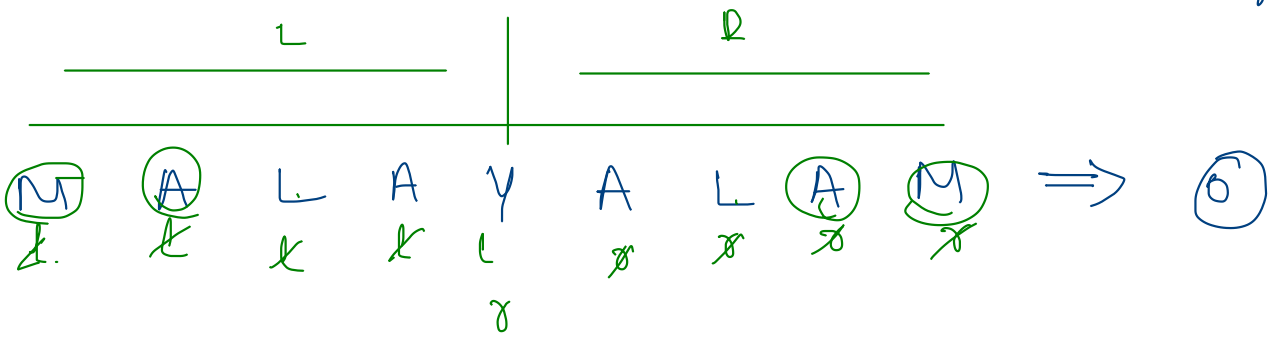


String s = "x y z m x z y"

x y z m z x y

→ x y z m z y x ⇒ palindrome

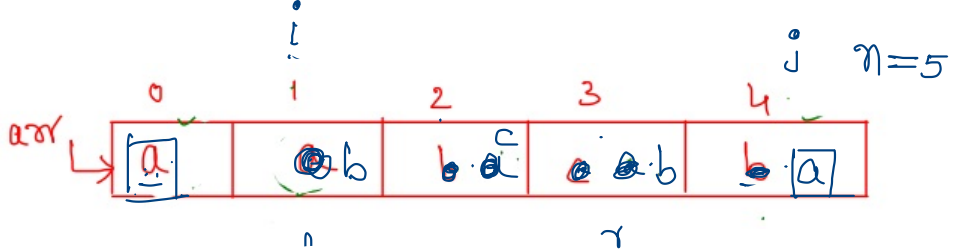
← 2-swaps



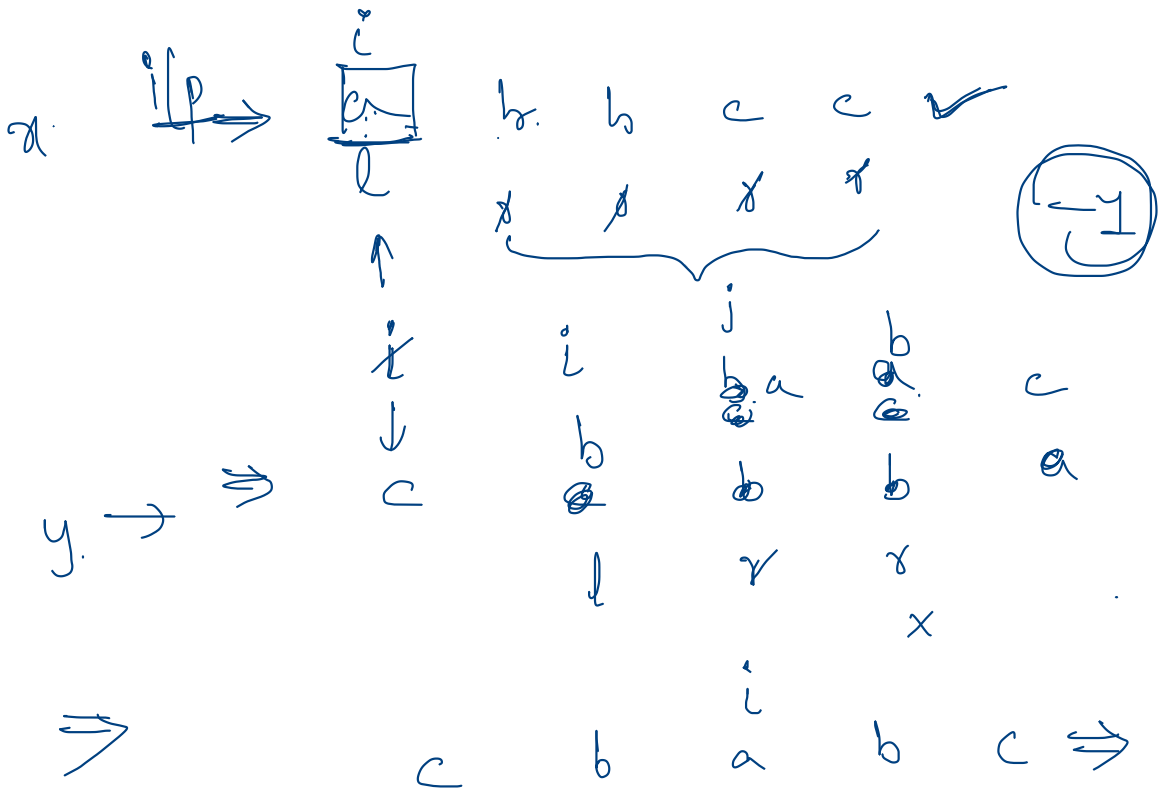
AMMA

Ex:- AMAM → 1

AMMA



Adj



count = 0 1 2 3

```
function minAdjSwaps(arr[],n)
{
    count=0
    for(i=0;i<n/2;i++)
    {
        left=i
        right=n-left-1
        while(left<right)
        {
            if(arr[left]==arr[right])
                break;
            else
                right--
        }
        if(left==right)
            return -1
        else
        {
            for(j=right;j<n-left-1;j++)
            {
                swap(arr[j],arr[j+1])
                count++
            }
        }
    }
    return count
}
```

Step 1 :-

x = minAdjSwaps(s)

y = minAdjSwaps(rev(s))

```
if(x>y)
{
    print(x)
}
else
{
    print(y)
}
```

Rotate Elements

Description

Given a n by n matrix. You have to rotate the elements of each ring of the matrix in the clockwise direction one place.

Input

Input Format

First line will contain a single number n

Next n lines will contain the matrix

Constraints

n<=1000

Elements of the matrix <=10000

Output

You have to display the rotated matrix

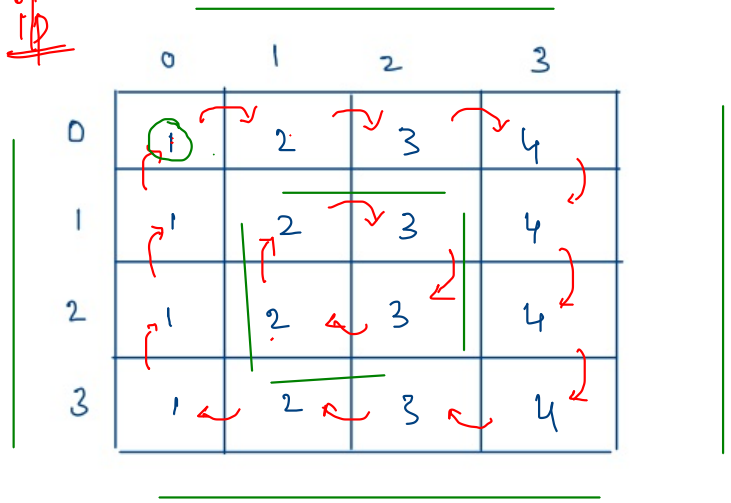
Sample Input 1

```
4
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
```

Sample Output 1

```
1 1 2 3
1 2 2 4
1 3 3 4
2 3 4 4
```

ip



⇒

op

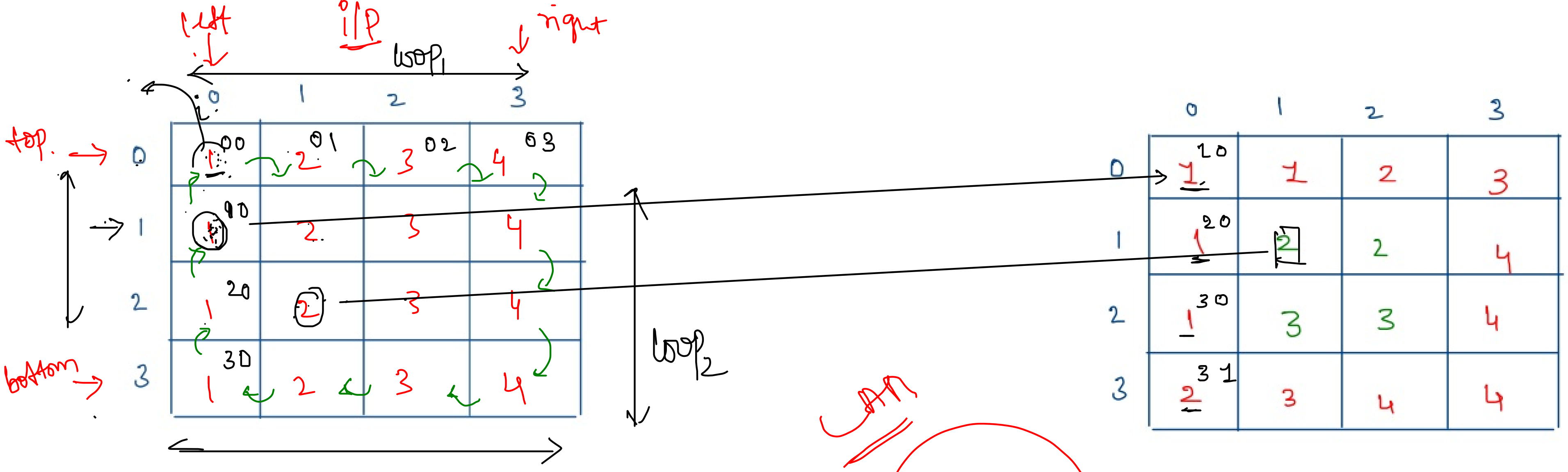
	0	1	2	3
0	1	1	2	3
1	1	2	2	4
2	1	3	3	4
3	2	3	4	4

{spiral . T}

print( )

```
4
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
```

```
1 1 2 3
1 2 2 4
1 3 3 4
2 3 4 4
```




```
function fun(arr[][],n)
{
        top=0,bottom=n-1,left=0,right=n-1
        while(left<right && top<bottom)
        {
                loop1
        top++
                loop2
        right--
                loop3
        bottom--
                loop4
        left++
        }
}
```

```
prev = arr[top+1][left]
for(i=left; i<=right; i++)
{
    temp = arr[top][i]
    arr[top][i] = prev
}   prev = temp
```

EA

6 → 4

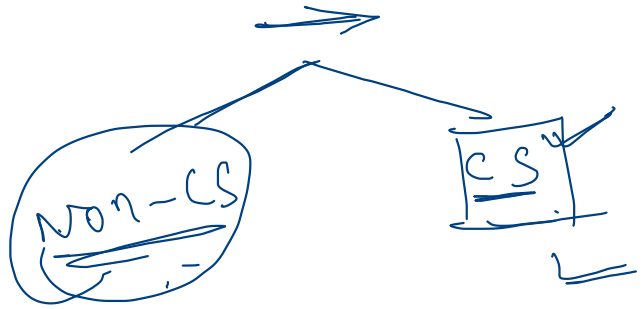
1 WT → DSA (3)

5 I

↳ 3 (DSA)

2 (DSA + project) M.Tech

↳ Python + ML



```
fun(n)
{
   if(n==0)
      return
   else
   {
         print(n)
         fun(n-1)
         fun(n-1)
    }
}
main()
{
    fun(3)
}
```

break

10:55

```
int fun(int p,int q)
{
    if(p<q)
       return 0
    else if(p==q)
       return p+fun(p-1,q)
    else
       return q+fun(p-2,q)
}
main()
{
     print(fun(7,5))
}
```

```
int p(int a)
{
     if(a==0)
         return 1
     else
         return p(a-1) + q(a-1)
}
int q(int a)
{
     if(a==0)
          return 2
     else
          return q(a-1)+q(a-1)
}

main()
{
    print( p(q(3))
}
```