iNautix Test -14

1. Haloween Party

Lenny is attending a Halloween party with his girlfriend Penny. At the party, Penny spots a giant rectangular chocolate bar. If the chocolate can be served as only 1 x 1 sized pieces, and Lenny can cut the chocolate bar exactly **K** times, what is the maximum number of chocolate pieces Lenny can cut and give Penny?

**Note**  
Chocolate must be served in size of 1 x 1 size pieces.  
Leonard can't move any of the pieces, including placing any piece on top of other.

Complete the max\_Chocolates function in your editor. It has 1 parameter:

  1. An integer array arr, where ai denotes the ith test case.

It must return an array, where ith element in the array denotes the result for the ith test case.

**Input Format**  
The locked stub code in your editor reads the following input from stdin and passes it to your function:

The first line contains an integer ***T***, the number of test cases. **T** lines follow.Each line contains an integer **K.**

**Constraints**  
1 ≤ T ≤ 10  
2 ≤ K ≤ 107

**Output Format**  
Your function must return an array, where ith element in the array denotes the result for the ith test case.

This is printed to stdout by the locked stub code in your editor.

**Sample Input 1**

4

5

6

7

8

**Sample Output 1**

6

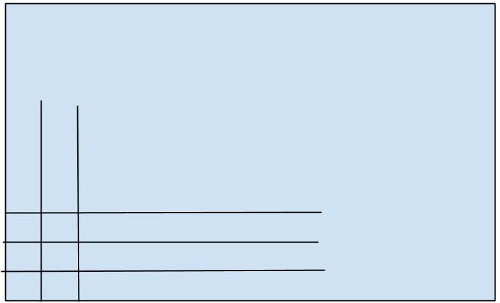
9

12

16

**Explanation**

Sample Case 1: The explanation below is for the first two test-cases. The rest of them follow a similar logic.

For the first test-case where K = 5, You need 3 Horizontal and 2 vertical cuts.  


For the second test-case where K = 6, You need 3 Horizontal and 3 vertical cuts.

Solution:

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main() {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

int t;

scanf("%d",&t);

while(t--)

{

long long int n,a,b;

scanf("%d",&n);

if(n%2==0)

{

a=n/2;

b=n/2;

}

else

{

a=n/2;

b=(n-a);

}

printf("%lld\n",a\*b);

}

return 0;

}

1. Grid Game

Julia is playing a game on an infinite 2-dimensional grid with the bottom left cell referenced as *(1, 1)*. All the cells contain a value of zero initially. The game consists of *n* steps. In each step, Julia is given two integers *a* and *b*. The value of each of the cells in the coordinate *(u, v)* satisfying *1 ≤ u ≤ a* and *1 ≤ v ≤ b*, is increased by *1*. After *n* such steps, if *x* is the largest number in any cell on the board, how many instances of *x*are there on the board?

Complete the function *countX* that has one parameter, a string array, *steps*, denoting the values of a and b for each of steps of the game. The function should return the total number of occurrences of greatest integer x in the grid after n steps.

**Input Format**  
The first line of input contains a single integer *n*. Each of the next n lines contains two space-separated integers, *a* and *b*.

**Constraints**

* *1 ≤****n****≤ 100*
* *1 ≤****a, b****≤ 10*

**Output Format**

The function should return the total number of occurrences of greatest integer x in the grid after n steps.

**Sample Input**

3

2 3

3 7

4 1

**Sample Output**

2

**Explanation**

Assume that the following board corresponds to cells (i, j) where 1 ≤ i ≤ 4 and 1 ≤ j ≤ 7.

At the beginning the board is in the following state:

0 0 0 0 0 0 0

0 0 0 0 0 0 0

0 0 0 0 0 0 0

0 0 0 0 0 0 0

After the first step we obtain:

0 0 0 0 0 0 0

0 0 0 0 0 0 0

1 1 1 0 0 0 0

1 1 1 0 0 0 0

After the second step we have:

0 0 0 0 0 0 0

1 1 1 1 1 1 1

2 2 2 1 1 1 1

2 2 2 1 1 1 1

Finally, after the last step the board will look like this:

1 0 0 0 0 0 0

2 1 1 1 1 1 1

3 2 2 1 1 1 1

3 2 2 1 1 1 1

So, the maximum number is 3 and there are exactly two cells which contain 3. Hence the answer is 2.

Solution:

/\*  
 \* Complete the function below.  
 \*/

Int main()

{

Int n,i;

Scanf(“%d”,&n);

Char \*\*steps=(char \*\*)malloc(n \*sizeof(char \*));

For(i=0;i<n;i++)

{

Steps[i]=(char \*)malloc(10 \*sizeof (char));

Scanf(“%s”,steps[i]);

}

Long int ans;

Ans=countX(n,steps);

Printf(“%d”,ans);

Return 0;

}

long countX(int steps\_size, char\*\* steps) {  
      
    char \*token;  
    long int ans=0;  
    int i,x=0,rows[1000],col[1000];  
      
    for(i=0;i<steps\_size;i++) // loop to break string into integers using strtok and atoi  
        {  
       // printf("%s\n",steps[i]);  
        int k=0,a[3],y=0;  
        char \*space=" ";  
        token=strtok(steps[i],space);  
        while(token!=NULL)  
        {  
            a[k]=atoi(token);  
            token=strtok(NULL,space);  
            k++;  
        }  
       // printf("%d  %d  \n",a[0],a[1]);  
        rows[x]=a[0];  
        col[x]=a[1];  
        x++;  
    }  
    int row\_max=rows[0]; //find maximum rows  
    for(i=0;i<x;i++)  
    {  
        if(rows[i]>row\_max)  
            row\_max=rows[i];  
    }  
    int col\_max=col[0];  
    for(i=0;i<x;i++)  
    {  
        if(col[i]>col\_max) //find maximum columns  
            col\_max=col[i];  
    }  
 //   printf("%d %d ",row\_max,col\_max);  
    int arr[1000][1000],j,var=0;  
    for(i=row\_max;i>=1;i--)  
    {  
        for(j=col\_max;j>=1;j--)  
        {  
            arr[i][j]=0; //create a matrix of max row max col size and initialize to 0  
        }  
    }  
    while(steps\_size--)  
    {  
    for(i=rows[var];i>=1;i--)  
        {  
        for(j=col[var];j>=1;j--)  
        {  
            arr[i][j]++; // increment the matrix values between indices a and b by 1  
        }  
    }  
        var++;  
    }    long long int max=arr[1][1];  
    printf("%d ",arr[0][0]);  
    for(i=1;i<=row\_max;i++)  
        {  
        for(j=1;j<=col\_max;j++)  
            {  
            if(arr[i][j]>max)  
                {  
                max=arr[i][j];             //find maximum value in the matrix

}  
        }  
    }  
   for(i=1;i<=row\_max;i++)  
       {  
       for(j=1;j<=col\_max;j++)  
           {  
           if(arr[i][j]==max) //find no of occurances of max value in matrix  
  
               ans++;  
       }  
   }  
                     
return ans;  
}