Arrays - Programming

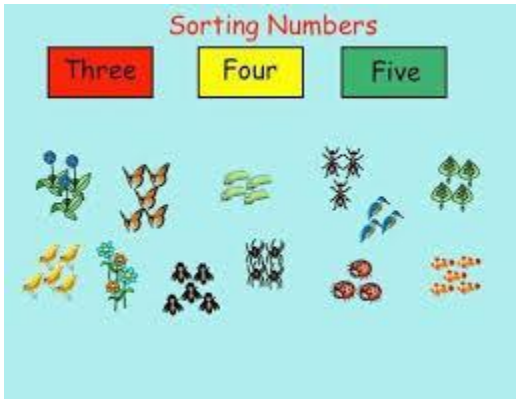
Problem-1

Sort by Shift 1

Sort by Shift 1



One day, Mahirl is interested in finding how to sort a sequence of integers a1, a2, ... an in non-decreasing order.



Being a young kid, the only operation she can perform is a unit shift. That is, she can move the last element of the sequence to its beginning:

a1, a2, ..., an → an, a1, a2, ..., an - 1.

Help Mahirl to calculate: What is the minimum number of operations that she needs to sort the sequence?

Input and Output Format:

The first line contains an integer, n ($2 \leq n \leq 105$).

The second line contains n integer numbers a1, a2, ..., an ($1 \leq ai \leq 105$).

If it's impossible to sort the sequence output -1. Otherwise output the minimum number of operations Mahirl needs to sort it.

Sample Input 1

2

2 1

Sample Output 1

1

Sample Input 2

3

1 3 2

Sample Output 2

-1

Sample Input 3

2

1 2

Sample Output 3

0

Solution :

```
#include<stdio.h>
int main()
{
    int n,i,a[100],p=0,j,k,temp=0;
    scanf("%d ",&n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(i=0;i<n-1;i++)
    {
        if(a[i]>a[i+1])
        break;
    }
    if(i==n-1)
    {
    printf("0");
    }
    else
    {
        for(k=0;k<n;k++)
        {
            temp=a[n-1];
            p++;
            for(j=n-1;j>0;j--)
            {
                a[j]=a[j-1];
            }
            if(j==0)
            a[0]=temp;
            for(i=0;i<n-1;i++)
            {
               if(a[i]>a[i+1])
                break;
            }
            if(i==n-1)
            {
                printf("%d",p);
                break;
            }
        }
    }
    if(k==n)
    printf("-1");
    return 0;
}
```

Problem-2

Rita and Children

Rita and Children

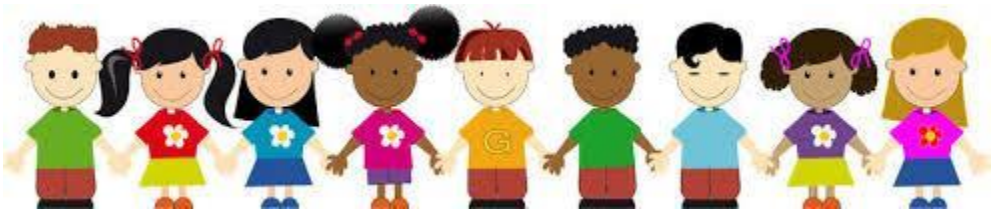


There are n children in Rita's school. Rita is going to give some candies to them. Let's number all the children from 1 to n. The i-th child wants to get at least ai candies.





Rita asks children to line up. Initially, the i-th child stands at the i-th place of the line. Then Rita starts distribution of the candies.



She follows the algorithm:

1. Give m candies to the first child in the line.
2. If this child still haven't got enough candies, then the child goes to the end of the line, else the child goes home.
3. Repeat the first two steps while the line is not empty.

Consider all the children in the order they go home. Rita wants to know, which child will be the last in this order?





Input and Output Format

The first line contains two integers n, m ($1 \le n \le 100$; $1 \le m \le 100$). The second line contains n integers a1, a2, ..., an ($1 \le ai \le 100$).

Output a single integer, representing the number of the last child.

Sample Input 1

5 2

1 3 1 4 2

Sample Output 1

4

Sample Input 2

6 4

1 1 2 2 3 3

Sample Output 2

6

{**Hint :**

Let's consider the first sample.

First child 1 gets 2 candies and goes home. Then child 2 gets 2 candies and goes to the end of the line. Currently the line looks like [3, 4, 5, 2] (indices of the children in order of the line). Then child 3 gets 2 candies and goes home, and then child 4 gets 2 candies and goes to the end of the line. Currently the line looks like [5, 2, 4]. Then child 5 gets 2 candies and goes home. Then child 2 gets two candies and goes home, and finally child 4 gets 2 candies and goes home.

Child 4 is the last one who goes home.}

```
#include<stdio.h>

 int main()
```

```
{
int a[100],n,k,i,c=0,t=0,z=0;
scanf("%d%d",&n,&k);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<n;i++)
{
	if(a[i]>k)
	break;
}
if(i==n)
printf("%d",n);
else
{
	for(i=0;i<n;i++)
	{
		c=0;
		while(a[i]>0)
		{
		a[i]=a[i]-k;
		c++;
		}
```

```
                if(t<=c)
                {
                t=c;
                z=i;
                }
        }
        printf("%d",z+1);

}
return 0;
}
```

Problem-3

Queue

Bus Queue

The bus stop queue has "*n*" groups of people. The *i*-th group from the beginning has a_i people. Every 30 minutes an empty bus arrives at the bus stop, it can carry atmost *"m"* people. Naturally, the people from the first group enter the bus first. Then go the people from the second group and so on. Note that the order

of groups in the queue never changes. Furthermore, if a group cannot fit all of its members into the current bus, they must wait for the next bus along with other groups standing after it in the queue.

Your task is to determine how many buses are needed to transport all *n* groups to the Dacha countryside.

Input Format

The first line contains two integers n and m ($1 \le n,m \le 100$). The next line contains *n* integers: $a_1, a_2, ..., a_n$ ($1 \le a_i \le m$).

Output Format

Print a single integer — the number of buses that are needed to transport all *n* groups to the Dacha countryside.

Sample Input

4 3

2 3 2 1

Sample Output

3

Solution :

```
#include<stdio.h>
int main()
{
    int n,m,a[100],i,c=0,s;
    scanf("%d %d",&n,&m);
    for(i=0;i<n;i++)
    scanf("%d",&a[i]);
    for(i=0;i<n;i++)
    {
        s=m;
        while(s-a[i]>=a[i+1])
        {
            s=s-a[i];
            i++;
        }
        c++;
    }
    printf("%d",c);
    return 0;
}
```

Problem-4

Mahirl and Antique Items

Mahirl and Antique Items



Mahirl is a great admirer of antique items and she has a huge collection of antique items at home. Once she wanted to expand her collection with exactly one antique item.



Mahirl knows n sellers of antiques, the i-th of them auctioned ki items. Currently the auction price of the j-th object of the i-th seller is sij. Mahirl gets on well with each of the n sellers. She is perfectly sure that if she outbids the current price of one of the items in the auction (in other words, offers the seller the money that is strictly greater than the current price of the item at the auction), the seller of the object will immediately sign a contract with her.





Unfortunately, Mahirl has only v units of money. Help her to determine which of the n sellers she can make a deal with.



Input and Output Format:

The first line of the input contains two space-separated integers n, v ($1 \le n \le 50$; $v \le 1000000$) — the number of sellers and the units of money Mahirl has.

Then n lines follow. The i-th line first contains integer ki ($1 \le ki \le 50$) the number of items of the i-th seller. Then go ki space-separated integers si1, si2, ..., siki ($sij \le 1000000$) — the current prices of the items of the i-th seller.

In the first line of the output, print integer p — the number of sellers with whom Mahirl can make a deal.

In the second line print p space-separated integers q1, q2, ..., qp ($1 \le qi \le n$) — the numbers of the sellers with whom Mahirl can make a deal. Print the numbers of the sellers in the increasing order.

Sample Input 1

3 50000

1 40000

2 20000 60000

3 10000 70000 190000

Sample Output 1

3

1 2 3

Sample Input 2

3 50000

1 50000

3 100000 120000 110000

3 120000 110000 120000

Sample Output 2

0

{**Hint:**

In the first sample Mahirl can bargain with each of the sellers. She can outbid the following items: a 40000 item from the first seller, a 20000 item from the second seller, and a 10000 item from the third seller.

In the second sample Mahirl cannot make a deal with any of the sellers, as the prices of all items in the auction are too big for her.}

Solution :

```
#include<stdio.h>
int main()
{
    int n,a,i,j,h=0,c,g[50],k=0;
    long int v,b;
    scanf("%d %ld",&n,&v);
    for(i=0;i<n;i++)
    {
        c=0;
        scanf("%d",&a);
        for(j=0;j<a;j++)
        {
            scanf("%ld",&b);
            if(v>b)
            c++;
        }
        if(c>0)
        {g[k++]=i+1;
        h++;}
    }
    if(h==0)
    printf("0");
    else
    {
        printf("%d\n",h);
        for(i=0;i<k;i++)
        printf("%d ",g[i]);
    }
    return 0;
}
```

Problem-5

Centroid TCS

Centroid

Write a program to find the centroid of an object in a 2-D grid.

A 2D grid is represented by values '0'(zeros) for background and by values '1'(ones) representing the object. Assume that the coordinates of the 1st point in the grid is (0,0). Assume that the 2D grid always consists of only one object surrounded by zeros as given below.

The centroid is a float value pair (xc,yc) where xc is the average of all x coordinate values of all the coordinates belonging to the object and yc is the average of all y coordinate values of all the coordinates belonging to the object.

Input and Output Format:

Input consists of (m*n) + 2 integers.

The 1st 2 integers are on 2 separate lines.

The 1st integer corresponds to m, the number of rows in the grid.

The 2nd integer corresponds to n, the number of columns in the grid.

The next 'n' integers correspond to the values in the 1st row.

The next 'n' integers correspond to the values in the 2nd row and so on.

Output consists of 2 float values that correspond to the x and y coordinate of the centroid. The 2 float values are separated by a space. The float values are printed correct to 2 decimal places.

Sample Input:

8

8

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 1 1 1 1 0 0
0 0 0 1 1 0 0 0
0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

Sample Output:

3.16,3.50

```
i=int(input())
j=int(input())
y=[]
t=[]
r=[]
c=0
for a in range (i):
    r.append((input().split()))
for p in range(i):
    for h in range(j):
        if(r[p][h]=="1"):
            y.append(h)
            t.append(p)
            c=c+1
u=round(float(sum(t)/c),2)
s=round(float(sum(y)/c),2)
print("%.2f"% u,end="")
print(",",end="")
print("%.2f"% s,end="")
```

Problem-6

Chess Board 1

Chess Board 1



Given the position of a bishop and a queen in a n*n chessboard, mark the remaining positions in the chess board as follows:

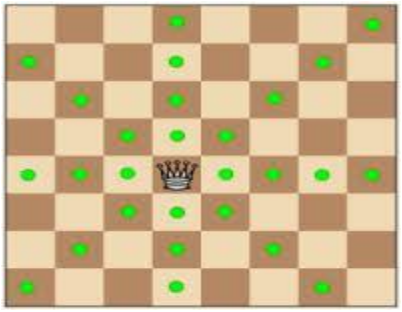
'*' --- if it is under attack from bishop

'$' --- if it is under attack from queen

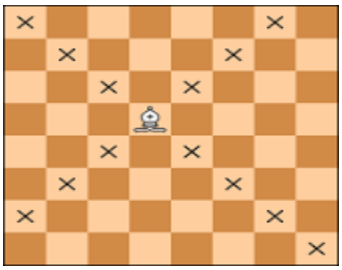
'%' --- if it is under attack from both queen and bishop.

'.' --- if it is not under attack.









Input and Output Format:

Input consists of 5 integers where first integer, n, corresponds to the size of the chess board.

Second and third integers correspond to the x and y coordinates of the bishop respectively, and fourth and fifth integers correspond to the x and y coordinates of the queen respectively.

Output consists of a *nxn* matrix obtained by applying the above rules.

Sample Input:

4

2

2

3

4

Sample Output:

*$*$

.B$$

%$%Q

..$%

SOLUTION

```
#include<stdio.h>
int main()
{
    int n,i,j,x1,y1,x2,y2,a[50][50],xd=0,xa=0,ya,yd;
    scanf("%d",&n);
    scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
    x1=x1-1;
    y1=y1-1;
    xd=x1-y1;
    xa=x1+y1;
    x2=x2-1;
    y2=y2-1;
    yd=x2-y2;
    ya=x2+y2;
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            if(i==x1&&j==y1)
            a[i][j]=66;
```

```
            else if(i==x2&&j==y2)
            a[i][j]=81;
            else if((i-j==xd||i+j==xa)&&(i-j==yd||i+j==ya||i==x2||j==y2))
            a[i][j]=37;
            else if(i-j==xd||i+j==xa)
            a[i][j]=42;
            else if(i==x2||j==y2||i-j==yd||i+j==ya)
            a[i][j]=36;
            else
            a[i][j]=46;
        }
    }
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            printf("%c",a[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Problem-7

Image Enlarger

Image Enlarger



Problem Statement

Given an image as a 2-D character array or a string [], scale it up by factor. That is, if the original image is M x N, the scaled image should be (M*factor) x (N*factor). Each character in the input should be represented by factor x factor of the same character in the output (see examples).

Constraints

- Range of M and N will be between 1 and 50 elements, inclusive.

- Each character in image will have ASCII value between 32 and 126, inclusive.

- Factor will be between 0 and 50, inclusive.

- The output will contain at most 7500 characters.

Input and Output Format:

The first line of the input consists of an integer that corresponds to M.

The second line of the input consists of an integer that corresponds to N.

The next M lines of the input corresponds to the image.

The last line of the input consists of an integer that corresponds to the factor.

Output consists of the scaled up image.

Refer sample output for formatting specifications.

Sample Input:

3

3

abc

def

ghi

3

Sample Output:

aaabbbccc

aaabbbccc

aaabbbccc

dddeeefff

dddeeefff

dddeeefff

ggghhhiii

ggghhhiii

ggghhhiii

Solution

```
#include<stdio.h>
```

```
int main()
{
    int i,j,x,y,k,n,l;
    char a[100][100];
    scanf("%d %d",&x,&y);
    for(i=0;i<x;i++)
    {
        for(j=0;j<y;j++)
        {
            scanf(" %c",&a[i][j]);
        }
    }
    scanf("%d",&n);
    for(i=0;i<x;i++)
    {
        for(l=0;l<n;l++)
        {
            for(j=0;j<y;j++)
            {
                for(k=0;k<n;k++)
                {
                        printf("%c",a[i][j]);
                }
            }
            printf("\n");
        }
    }
    return 0;

}
```

Problem-8

Spikes Detection TCS

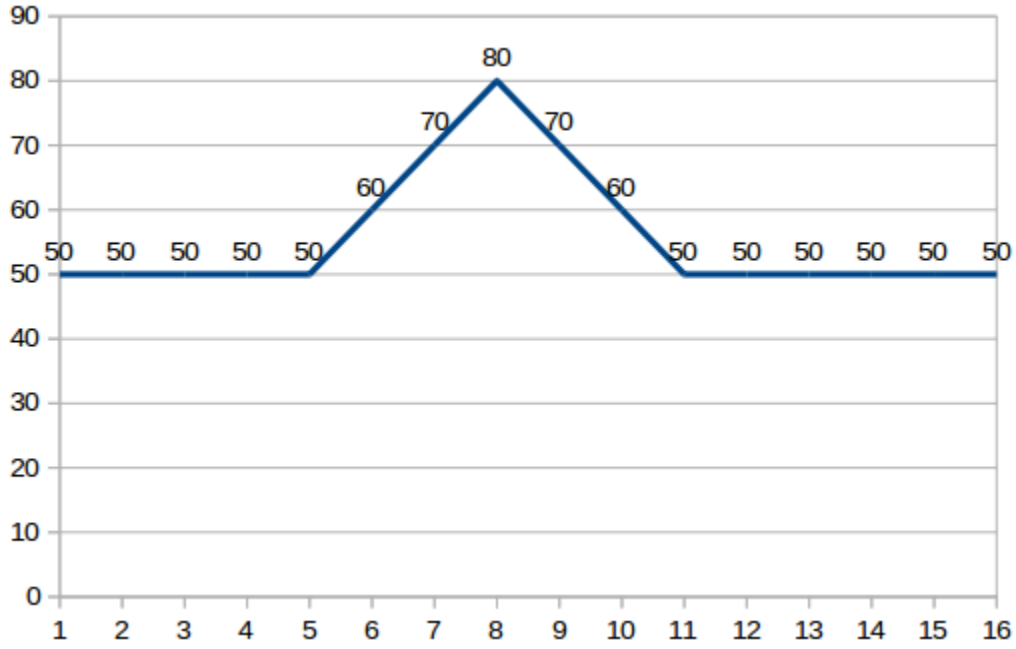
Spike Detection

Write a program to find the number of spikes, the origination of a spike and its span in a dc signal sample.

A spike in this case is assumed to be a segment where the signal increases to a point and then decreases (or vice versa). The point from where it starts increasing and decreasing (or vice versa) and back to the stable point is said to be the span of spike.

Example :

50 50 50 50 50 60 70 80 70 60 50 50 50 50 50 50



In the above sequence, there is 1 spike, the spike starts at 5 and the span is 7.

Input and Output Format:

Input consists of n+1 integers.

The 1st integer corresponds to n, the number of signal sample values.
The next n integers correspond to the signal sample values in order.

Output consists of the number of spikes and the origin and span of each spike.
Assume that the maximum number of spikes in the input signal is 20.
If there are multiple spikes, print the spike details in the order in which they appear in the signal.
Refer sample output for formatting specifications.

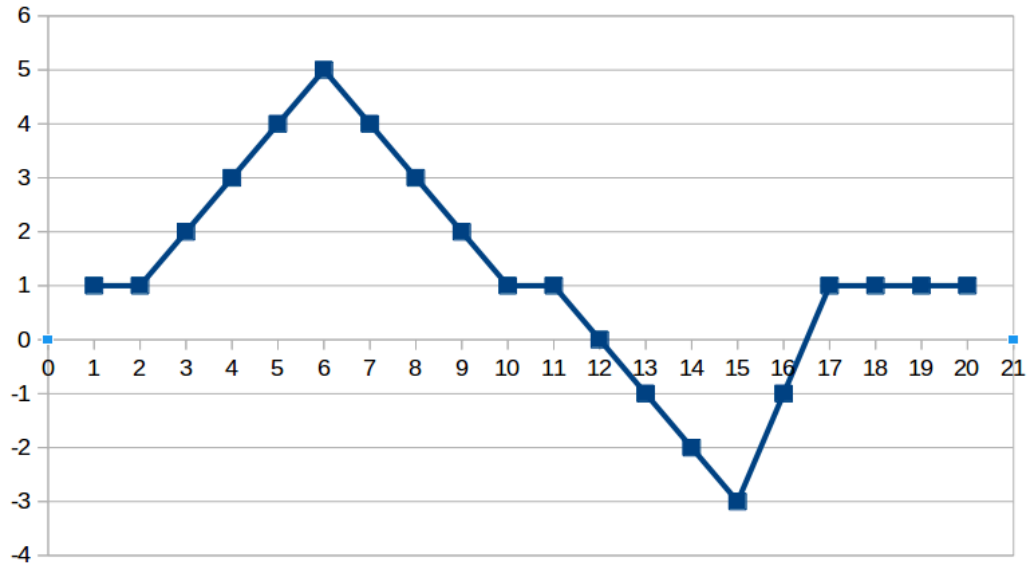
Sample Input 1:
16
50
50
50

50
50
60
70
80
70
60
50
50
50
50
50
50

Sample Output 1:
1
5 7

Sample Input 2:
20
1
1
2
3
4
5
4
3
2
1
1
0
-1
-2
-3
-1
1
1
1
1
Sample Output 2:
2
2 9
11 7

Explanation for sample 2:



Solution :

```
#include<stdio.h>
int main()
{
    int i,j,n,a[100],c=0,k,p=0,d[200],e[200];
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
for(j=0;j<n;j++)
    {
        if(a[j+1]!=a[j])
        {
            for(k=j+1;k<n;k++)
            {
               // printf("%d ",a[k]);
                if(a[k]==a[j])
                {
                    c++;
                    //printf(" /%d/%d/ ",j+1,k-j+1);

                    d[p]=j+1;
                    e[p]=k-j+1;
                    j=k-1;
                   p++;
                   break;
                }
            }
        }

    }
    printf("%d\n",c);
    for(i=0;i<p;i++)
    {
        printf("%d %d",d[i],e[i]);
```

```
        printf("\n");
    }
        return 0;

}
```

Problem-9

Land Mines

Land Mines

Problem Statement

We want to send a soldier into a mine field with a directional metal detector. The field is an n x n grid, and the soldier will move horizontally (east or west) or vertically (north or south), never leaving the field. His metal detector can be pointed in any of the cardinal directions (east, north, west, or south) and will beep if it senses any mine in that direction, no matter how far away the mine is. Only mines that are exactly in the row or column down which the sensor is pointed are sensed.

The soldier can move one square at a time in any of the four directions, pointing his sensor in various directions. But he will not move onto a square if his sensor beeps when pointed in that direction, unless he has previously visited that square. He will continue this process, visiting and revisiting squares as needed, until he is sure that he has visited every square that he can.

Write a program that takes the actual layout of mines in a minefield as input and that returns the number of squares that the soldier can safely visit.

The layout shows the mine field as a String[], starting with the top row. '-' denotes a square with no mine and 'M' denotes a square that contains a mine. The soldier will always start in the northwest corner (the leftmost square in the top row), which will never contain a mine.

Constraints

- layout contains between 2 and 50 elements inclusive.
- The length of each element of layout equals the number of elements in layout.
- Each character in layout will be a hyphen ('-') or uppercase 'M'.
- The first character in the first element of layout will be a hyphen.

Input and Output Format:

The first line of the input consists of an integer that corresponds to the size of the grid/ layout. The next n lines consists of the mine layout.

Output consists of a single integer that corresponds to the number of squares that the soldier can safely visit.

Sample Input 1:

3

```
-M-
---
M--
```

Sample Output 1:
1

Hint:
The soldier cannot safely move in either direction from his initial position.

Sample Input 2:
3
```
-M-
-M-
--M
```

Sample Output 2:
3

[Hint : The soldier can safely go south from his initial position, but cannot head east from any of those locations.]

Sample Input 3:
4
```
--M-
-MM-
----
----
```

Sample Output 3:
12

[Hint : The soldier can go south, and then can move east along both of the bottom rows. From the easternmost location on one of the bottom rows, he can safely move north. But the second square in the top row can never safely be visited.]

Sample Input 4:
5
```
-----
--M-M
-----
-M---
---M-
```

Sample Output 4:

[**Hint :** All of the squares without mines can safely be visited. For example, a way to visit the second square on the bottom row is as follows: go south 2, go east 2. From this location it is safe to go south since the detector indicates no mines in that direction; go south 2. Then go west 1.]

Solution :

Problem-10

Minesweeper TCS

MINESWEEPER



Have you ever played Minesweeper? It's a cute little game which comes within a certain Operating System which name we can't really remember. Well, the goal of the game is to find where are all the mines within a MxN field. To help you, the game shows a number in a square which tells you how many mines there are adjacent to that square. For instance, suppose the following 4x4 field with 2 mines (which are represented by an * character):

```
*...
....
.*..
....
```

If we would represent the same field placing the hint numbers described above, we would end up with:

```
*100
2210
1*10
1110
```

As you may have already noticed, each square may have at most 8 adjacent squares.

Input Format:

The first line of the input contains two integers n and m (0<n, m<=100) which stands for the number of lines and columns of the field. Each safe square is represented by a "." Character (without the quotes) and each mine is represented by a "*" character (without the quotes).

Output Format:

Output should be the field with the "." characters replaced by the number of adjacent mines to that square.
Refer sample input and output for formatting details.

Sample Input

4 4

```
*...
....
.*..
....
```

Sample Output

```
*100
2210
1*10
1110
```

Solution :

```
#include<stdio.h>
int main()
{
    int m,n,i,j,s[100][100];
    char c[100][100];
    scanf("%d%d",&m,&n);
    for(i=0;i<m;i++)
    {
        scanf("%s",c[i]);
    }
    for(i=0;i<m;i++)
    {
```

```
        for(j=0;j<n;j++)
        {
            if(c[i][j]=='*')
            s[i][j]=-1;
            else
            s[i][j]=0;
        }
    }
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            if(s[i][j]==-1)
            {
                if(s[i-1][j-1]!=-1)
                s[i-1][j-1]+=1;
                if(s[i-1][j]!=-1)
                s[i-1][j]+=1;
                if(s[i-1][j+1]!=-1)
                s[i-1][j+1]+=1;
                if(s[i][j-1]!=-1)
                s[i][j-1]+=1;
                if(s[i][j+1]!=-1)
                s[i][j+1]+=1;
                if(s[i+1][j-1]!=-1)
                s[i+1][j-1]+=1;
                if(s[i+1][j]!=-1)
                s[i+1][j]+=1;
                if(s[i+1][j+1]!=-1)
                s[i+1][j+1]+=1;
            }
        }
    }
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            if(s[i][j]==-1)
            {
                c[i][j]='*';
                printf("%c",c[i][j]);
            }
            else
            printf("%d",s[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```