≠≠ 2D ARRAY ≠≠

2d Arrays Demo

Easy

1. You are given a number n, representing the number of rows.
2. You are given a number m, representing the number of columns.
3. You are given n*m numbers, representing elements of 2d array a.
4. You are required to display the contents of 2d array as suggested by output format below.

Constraints

```
1 <= n <= 10^2
1 <= m <= 10^2
-10^9 <= e1, e2, .. n * m elements <= 10^9
```

Format

Input

A number n
A number m
e11
e12..
e21
e22..
.. n * m number of elements

Output

e11 e12 e13 ..
e21 e22 e23 ..

Example

Sample Input

```
2
```

```
4
11
12
13
14
21
22
23
24
```

Sample Output

```
11 12 13 14
21 22 23 24
```

```
#include<iostream>
#include<vector>
using namespace std;
int main(){


    //write your code here
    int r{};
    int c{};
    cin>>r;
    cin>>c;
    // int array[r][c];
    // //cout <<"fill the array";
    // for(int i{};i<r;i++){
    //     for(int j{};j<c;j++){
    //         cin>>array[i][j];
    //     }
    // }
    // for(int i{};i<r;i++){
    //     for(int j{};j<c;j++){
    //         cout<<array[i][j]<<" ";
    //     }
    //     cout<<endl;
    // }


vector<vector<int>> array;
    for(int i{};i<r;i++){
        vector<int> sarr;

        for(int j{};j<c;j++){
            int ele{};
            cin>>ele;
            sarr.push_back(ele);
        }
        array.push_back(sarr);
    }

    for(int i{};i<r;i++){
        for(int j{};j<c;j++){
            cout<<array.at(i).at(j)<<" ";
        }
        cout<<endl;
    }

    return 0;
}
```

Matrix Multiplication

Easy

1. You are given a number n1, representing the number of rows of 1st matrix.
2. You are given a number m1, representing the number of columns of 1st matrix.
3. You are given n1*m1 numbers, representing elements of 2d array a1.
4. You are given a number n2, representing the number of rows of 2nd matrix.
5. You are given a number m2, representing the number of columns of 2nd matrix.
6. You are given n2*m2 numbers, representing elements of 2d array a2.
7. If the two arrays representing two matrices of dimensions n1 * m1 and n2 * m2 can be multiplied, display the contents of product array as specified in output format.
8. If the two arrays can't be multiplied, print "Invalid input".

Constraints

1 <= n1 <= 10^2
1 <= m1 <= 10^2
-10^9 <= e11, e12, .. n1 * m1 elements <= 10^9
1 <= n2 <= 10^2
1 <= m2 <= 10^2
-10^9 <= e11', e12', .. n2 * m2 elements <= 10^9

Format

Input

A number n1
A number m1
e11 e12.. e21 e22.. .. n1 * m1 number of elements of array a1
A number n2
A number m2
e11' e12'.. e21' e22'.. .. n2 * m2 number of elements of array a2

Output

e11" e12" e13" .. e21" e22" e23" .. .. elements of prd array

Example

Sample Input

```
2
3
10
0
0
0
20
0
3
4
1
0
1
0
0
1
1
2
1
1
0
0
```

Sample Output

```
10 0 10 0
0 20 20 40
```

```
#include<iostream>
#include<vector>

using namespace std;

void matrixMult(vector<vector<int>> A, vector<vector<int>> B){
//     Write your code here.
int r1=A.size();
    int c1 =A[0].size();
    int r2=B.size();
    int c2=B[0].size();

    if(c1!=r2){
        cout <<"Invalid input"<<endl;
        return ;
    }else{
        vector<vector<int>>pdt;
        for(int i{};i<r1;i++){
            vector<int>arr;
            for(int j{};j<c2;j++){
                int val=0;
                for(int k{};k<c1;k++){
                    val+=A[i][k]*B[k][j];
                }
                arr.push_back(val);
            }
            pdt.push_back(arr);
        }
        for(int i{};i<r1;i++){
            for(int j{};j<c2;j++){
                cout<<pdt[i][j]<<" ";
            }
            cout<<endl;
        }
    }

}

int main(){
    int r1;
    int c1;
    cin>>r1;
    cin>>c1;

    vector<vector<int>> mat1;
    for(int i= 0; i< r1; i++){
        vector<int> arr;
        for(int j= 0; j< c1; j++){
            int ele;
            cin>> ele;
            arr.push_back(ele);
        }
        mat1.push_back(arr);
    }

    int r2;
    int c2;
    cin>>r2;
    cin>>c2;

    vector<vector<int>> mat2;
    for(int i= 0; i< r2; i++){
        vector<int> arr;
        for(int j= 0; j< c2; j++){
            int ele;
            cin>> ele;
            arr.push_back(ele);
        }
        mat2.push_back(arr);
    }

    matrixMult(mat1, mat2);
}
```

The State Of Wakanda - 1

Easy

The historic state of Wakanda has various monuments and souvenirs which are visited by many travelers every day. The guides follow a prescribed route of visiting the monuments which improve them understand the relevance of each monument.

The route of the monument is fixed and expressed in a 2-d matrix where the travelers visit the prescribed next monument. For example

1 2 3
4 5 6
7 8 9

is the prescribed route and the visitors travels this path: 1->2->3->4->5->6->7->8->9

However, a certain visitor decides to travel a different path as follows:
1. He first travels southwards till no further south places are available.
2. He then moves only 1 place eastwards.
3. He starts to move again towards north till any further north moves are available.
This continues till all the places are covered.

For example, the monuments are named as follows:
1 2 3
4 5 6
7 8 9

Path followed by traveler: 1->4->7->8->5->2->3->6->9

You are required to print the path that this traveler follows to visit all places.

1. You will be given a number n, representing the number of rows.
2. You will be given a number m, representing the number of columns.
3. You will be given n*m numbers, representing elements of 2d arrays.

Note - Please check the output format for details.

Constraints

```
1 <= n <= 10^2
1 <= m <= 10^2
-10^9 <= e1, e2, .. n * m elements <= 10^9
```

Format

Input

A number n
A number m
e11
e12..
e21
e22..
.. n * m number of elements

Output

e11
e12
e13
..
e1n
e2n
e2n-1
..
e21
e31
e32
..
e3n
e4n ..

Example

Sample Input

```
3
4
11
12
13
14
21
22
23
24
31
32
33
34
```

Sample Output

```
11
21
31
32
22
12
13
23
33
34
24
14
```

```
#include<iostream>
#include<vector>

using namespace std;

void columnTraversal(vector<vector<int>> mat){
//   Write your code here.
    int r = mat.size();
    int c = mat[0].size();
    for(int j{};j<c;j++){
        if (j%2!=0){
            for(int i{r-1};i>=0;i--){
                cout<<mat[i][j]<<endl;
            }
        }else{
            for(int i{};i<r;i++){
                cout<<mat[i][j]<<endl;
            }
        }
    }
}

int main(){
    int n;
    int m;
    cin>> n;
    cin>> m;

    vector<vector<int>> mat;
    for(int i= 0; i< n; i++){
        vector<int> arr;
        for(int j= 0; j< m; j++){
            int ele;
            cin>> ele;
            arr.push_back(ele);
        }
        mat.push_back(arr);
    }

    columnTraversal(mat);
}
```

Spiral Display

Easy

1. You are given a number n, representing the number of rows.
2. You are given a number m, representing the number of columns.
3. You are given n*m numbers, representing elements of 2d array a.
4. You are required to traverse and print the contents of the 2d array in form of a spiral.

Note - Please check the sample output for details.

Constraints

```
1 <= n <= 10^2
1 <= m <= 10^2
-10^9 <= e1, e2, .. n * m elements <= 10^9
```

Format

Input

A number n
A number m
e11

e12..
e21
e22..
.. n * m number of elements

Output

e11
e21
..
en1
en2
en3
..
enm
e(n-1)m
..
e1m
e1(m-1)
..
e12
e22
e32
..

Example

Sample Input

```
3
5
11
12
13
14
15
21
22
23
24
25
31
32
33
34
35
```

Sample Output

```
11
21
31
32
33
34
35
25
15
14
13
12
```

```
#include <iostream>
#include<vector>
using namespace std;

void spiral_traversal (vector<vector<int>> mat){
    int r = mat.size();
    int c = mat[0].size();
    int rx =r-1;
    int cx =c-1;

    int a{};
    int b{};

    int n{};

    int t=(r+1)/2;
    int q{1};
    while(q<=t){
        for(int i{a};i<=rx;i++){
            cout <<mat[i][b]<<endl;
            n++;
            if(n==r*c){
            return;
            }
        }
        //cout<<"aaaaaaa"<<endl;
        b++;
        for(int j{b};j<=cx;j++){
            cout <<mat[rx][j]<<endl;
            n++;
            if(n==r*c){
            return;
            }
        }
        rx--;
        //cout<<"aaaaaaa"<<endl;
        for(int i{rx};i>=a;i--){
            cout <<mat[i][cx]<<endl;
            n++;
            if(n==r*c){
            return;
            }
        }
        cx--;
        //cout<<"aaaaaaa"<<endl;
        for(int j{cx};j>=b;j--){
            cout <<mat[a][j]<<endl;
            n++;
            if(n==r*c){
                return;
                }
            }
            a++;
            //cout<<"aaaaaaa"<<endl;


            q++;
        }
}



int main(){
    //cout<<"Enter the size of 2d array: ";
    int r{};
    int c{};
    cin>>r;
    cin>>c;

    //cout <<"fill the array";


    vector<vector<int>> matrix;
    for(int i{};i<r;i++){
        vector<int> sarr;

        for(int j{};j<c;j++){
            int ele{};
            cin>>ele;
            sarr.push_back(ele);
        }
        matrix.push_back(sarr);
    }

    spiral_traversal(matrix);

    return 0;
}
```

Exit Point Of A Matrix

Easy

1. You are given a number n, representing the number of rows.
2. You are given a number m, representing the number of columns.
3. You are given n*m numbers (1's and 0's), representing elements of 2d array a.
4. Consider this array a maze and a player enters from top-left corner in east direction.
5. The player moves in the same direction as long as he meets '0'. On seeing a 1, he takes a 90 deg right turn.
6. You are required to print the indices in (row, col) format of the point from where you exit the matrix.

Constraints

```
1 <= n <= 10^2
1 <= m <= 10^2
e1, e2, .. n * m elements belongs to the set (0, 1)
```

Format

Input

A number n

A number m

e11

e12..

e21

e22..

.. n * m number of elements

Output

row

col (of the point of exit)

Example

Sample Input

```
4
4
0
0
1
0
1
0
0
0
0
0
0
0
1
0
1
0
```

Sample Output

```
1
3
```

```
#include <iostream>
#include<vector>
using namespace std;

void exit_points (vector<vector<int>> mat){
    int r = mat.size();
    int c = mat[0].size();

    int i{};
    int j{};

    int dir{};

    while( i!=-1 && i!=r && j!=-1 && j!=c ){
        //changing direction if required
        if(mat[i][j]==1){
            dir++;
            dir %= 4;
        }

        //moving one step ahead in the current direction
        if(dir==0){
            j++;
        }else if(dir==1){
            i++;
        }else if(dir==2){
            j--;
        }else if(dir==3){
            i--;
        }

    }

    if(i==-1){
        cout<<i+1<<endl<<j<<endl;
    }else if(i==r){
        cout<<i-1<<endl<<j<<endl;
    }else if(j==-1){
        cout<<i<<endl<<j+1<<endl;
    }else if (j==c){
        cout<<i<<endl<<j-1<<endl;
    }
}



int main(){
    //cout<<"Enter the size of 2d array: ";
    int r{};
    int c{};
    cin>>r;
    cin>>c;

   // cout <<"fill the array";


    vector<vector<int>> array;
    for(int i{};i<r;i++){
        vector<int> sarr;

        for(int j{};j<c;j++){
            int ele{};
            cin>>ele;
            sarr.push_back(ele);
        }
        array.push_back(sarr);
    }

    exit_points(array);

    return 0;
}
```

Rotate By 90 Degree

Easy

1. You are given a number n, representing the number of rows and number of columns.
2. You are given n*n numbers, representing elements of 2d array a.
3. You are required to rotate the matrix by 90 degree clockwise and then display the contents using display function.

Note - you are required to do it in-place i.e. no extra space should be used to achieve it .

Constraints

```
1 <= n <= 10^2
-10^9 <= e1, e2, .. n * n elements <= 10^9
```

Format

Input

A number n
e11
e12..
e21
e22..
.. n * n number of elements

Output

output is taken care of by display function

Example

Sample Input

```
4
11
12
13
14
21
22
23
24
31
32
33
34
41
42
43
44
```

Sample Output

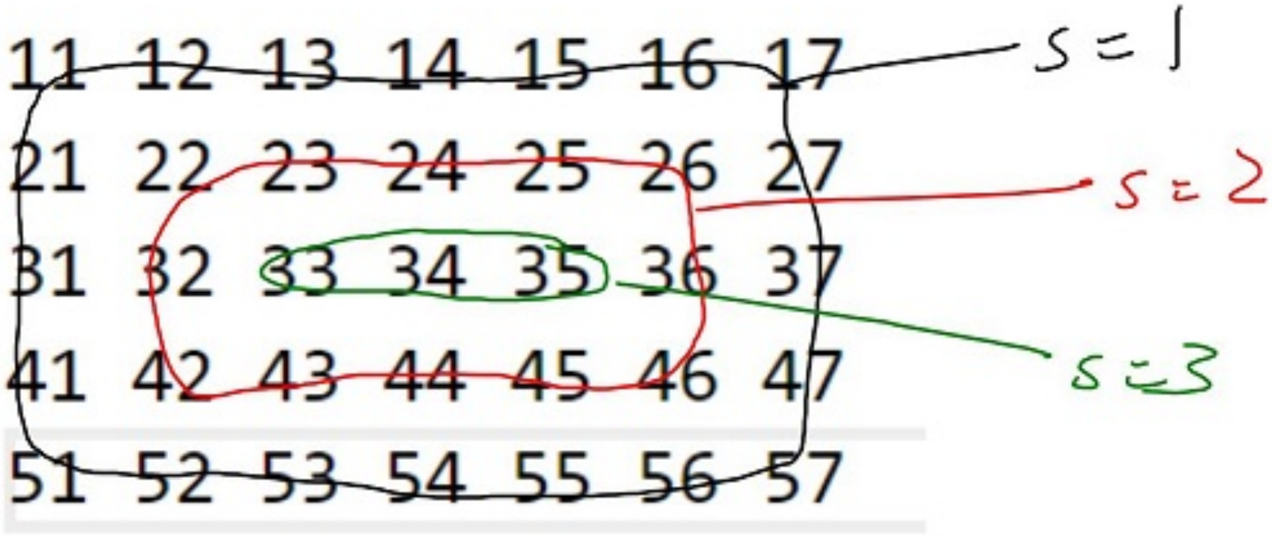
```
41 31 21 11
42 32 22 12
43 33 23 13
```

```
44 34 24 14
#include <iostream>
#include <vector>

using namespace std;

void rotate_and_display(vector<vector<int>> mat){
    int o = mat.size();
    //rotating the matrix

    ////making transpose first
    int temp{};
    for (int i{};i<o;i++){
        for (int j{i};j<o;j++){
            temp =mat[i][j];
            mat[i][j]=mat[j][i];
            mat[j][i]=temp;
        }
    }
    ////then reversing the order of column
    for (int i{};i<o;i++){
        int ci{0};
        int cx{o-1};
        int temp{};

        while(ci<cx){
            temp = mat[i][ci];
            mat[i][ci] = mat[i][cx];
            mat[i][cx] = temp;
            ci++;
            cx--;
        }
    }


    // //making new rotated matrix
    // vector<vector<int>> rmat;
    // for(int j{0};j<o;j++){
    //     vector<int>line;
    //     for(int i{o-1};i>=0;i--){
    //         int ele{};
    //         ele=mat[i][j];
    //         line.push_back(ele);
    //     }
    //     rmat.push_back(line);
    // }
    // cout<<"hello"<<endl;
    //displaying the matrix
    for(int i{};i<o;i++){
        for(int j{};j<o;j++){
            cout<<mat[i][j]<<" ";
        }
        cout<<endl;
    }

}



int main(){
    //creating the square matrix
    int o{};
    cin>>o;

    vector<vector<int>> matrix;
    for(int i{};i<o;i++){
        vector<int> line;
        for(int j{};j<o;j++){
            int ele{};
            cin>>ele;
            line.push_back(ele);
        }
        matrix.push_back(line);
    }

    rotate_and_display(matrix);
}
```

Ring Rotate

Easy

You are given a n*m matrix where n are the number of rows and m are the number of columns. You are also given n*m numbers representing the elements of the matrix.
You will be given a ring number 's' representing the ring of the matrix. For details, refer to image.



You will be given a number 'r' representing number of rotations in an anti-clockwise manner of the specified ring.
You are required to rotate the 's'th ring by 'r' rotations and display the rotated matrix.

Constraints

```
1 <= n <= 10^2
1 <= m <= 10^2
-10^9 <= e11, e12, .. n * m elements <= 10^9
0 < s <= min(n, m) / 2
-10^9 <= r <= 10^9
```

Format

Input

A number n
A number m
e11
e12..
e21
e22..
.. n * m number of elements of array a
A number s
A number r

Output

output is handled by display function

Example

Sample Input

```
5
7
11
12
13
14
15
16
17
21
22
23
24
25
26
27
31
32
33
34
35
36
37
41
42
43
44
45
46
47
51
52
53
54
55
56
57
2
3
```

Sample Output

```
11 12 13 14 15 16 17
21 25 26 36 46 45 27
31 24 33 34 35 44 37
41 23 22 32 42 43 47
51 52 53 54 55 56 57
```

```
#include <iostream>
#include<vector>
using namespace std;
vector <int> get_1d_array( vector<vector<int>>& mat,int s){
    int r = mat.size();
    int c = mat[0].size();

    int rmin{s-1};
    int cmin{s-1};
    int rmax{r-s};
    int cmax{c-s};
    vector <int> oned;
    for(int i{rmin};i<rmax;i++){
        oned.push_back(mat[i][cmin]);
    }
    for(int j{cmin};j<(cmax);j++){
        oned.push_back(mat[rmax][j]);
    }
    for(int i{rmax};i>rmin;i--){
        oned.push_back(mat[i][cmax]);
    }
    for(int j{cmax};j>cmin;j--){
        oned.push_back(mat[rmin][j]);
    }


    return oned;
}
void rotate_array(vector<int> &oned,int t){
    int n = oned.size();
    //t -=(2*t);//changing its sign because sense of rotation is
different in my solution (ahead)
    t = t%n ;

    if(t<0){
        t = n+t;
    }

    vector<int> temp;
    for(int i{};i<n;i++){
        temp.push_back(oned[i]);
    }

    for(int i{};i<n;i++){
        oned[(i+t)%n]= temp[i];
    }
}


void replace_shell ( vector<vector<int>>& mat,vector <int>
&oned ,int s){
    int r = mat.size();
    int c = mat[0].size();

    int rmin{s-1};
    int cmin{s-1};
    int rmax{r-s};
    int cmax{c-s};

    int n{};
    for(int i{rmin};i<rmax;i++){
        mat[i][cmin] = oned[n];
        n++;
    }
    for(int j{cmin};j<(cmax);j++){
        mat[rmax][j] = oned[n];
        n++;
    }
    for(int i{rmax};i>rmin;i--){
        mat[i][cmax] = oned[n];
        n++;
    }
    for(int j{cmax};j>cmin;j--){
        mat[rmin][j] = oned[n];
        n++;
    }


}


void ring_rotate (vector<vector<int>>& mat,int s,int t){
    // int r = mat.size();
    // int c = mat[0].size();

    vector <int> oned = get_1d_array(mat,s );
    rotate_array(oned,t);
    replace_shell(mat,oned,s);




}


void display( vector<vector<int>> &mat){

    int r = mat.size();
    int c = mat[0].size();

    for(int i{};i<r;i++){
        for(int j{};j<c;j++){
            cout<<mat[i][j]<<" ";
        }
        cout<<endl;
    }
}


int main(){
    //cout<<"Enter the size of 2d array: ";
    int r{};
    int c{};
    cin>>r;
    cin>>c;

   // cout <<"fill the array";


    vector<vector<int>> matrix;
    for(int i{};i<r;i++){
        vector<int> sarr;

        for(int j{};j<c;j++){
            int ele{};
            cin>>ele;
            sarr.push_back(ele);
        }
        matrix.push_back(sarr);
    }

    //enter the shell you want to rotate
    int s{};
    cin>>s;
    //enter how many times you want to rotate it
    int t{};
    cin>>t;
    ring_rotate(matrix,s,t);
    display (matrix);
    return 0;
}
```

The State Of Wakanda - 2

Easy

The historic state of Wakanda has various monuments and souveniers which are visited by many travellers every day. The guides follow a prescribed route of visiting the monuments which improves them understand the relevance of each monument. The route of the monument is fixed and expressed in a 2-d matrix where the travellers visit the prescribed next monument. For example

1 2 3
4 5 6
7 8 9

is the prescribed route and the visitors travels this path: 1->2->3->4->5->6->7->8->9

However, a certain visitor decides to travel a different path as follows:

1. The visitor only plans to visit the upper diagonal triangle of the monument list.
2. The visitor travels diagonally till there are no more moves left in the current journey.
3. He then visits the adjacent monument to the first monument of current diagonal journey.
4. He continues the same path till all the monuments of the upper half have been travelled.

For Example:

The monuments are named as:

1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16

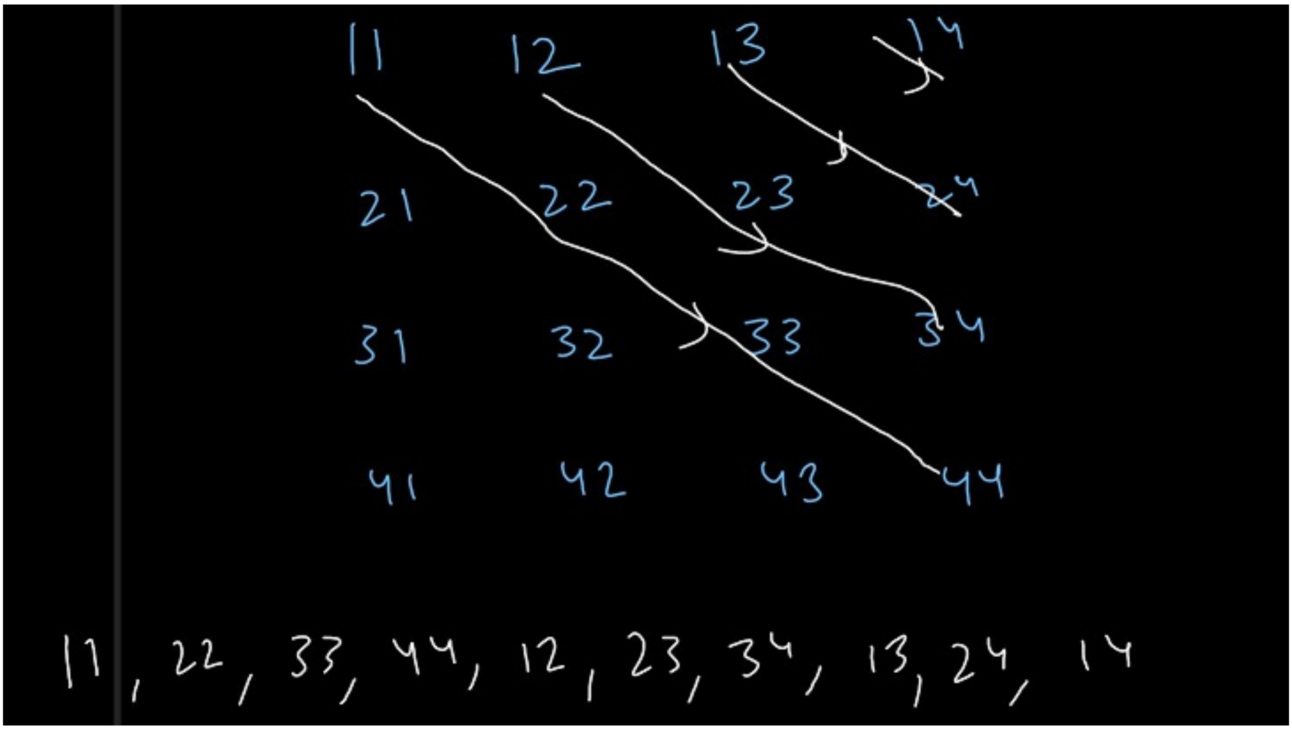
The path followed by the visitor is: 1->6->11->16->2->7->12->3->8->4

You are required to print the path followed by the traveller to visit all the monuments.

Refer to the photo for a better clarification.

1. You are given a number n, representing the number of rows and columns of a square matrix.
2. You are given n * n numbers, representing elements of 2d array a.
3. You are required to diagonally traverse the upper half of the matrix and print the contents.

For details check image.



Constraints

1 <= n <= 10^2
-10^9 <= e11, e12, .. n * m elements <= 10^9

Format

Input

A number n
e11
e12..
e21
e22..
.. n * n number of elements of array a

Output

Diagonal traversal as in image below

Example

Sample Input

```
4
11
12
13
14
21
22
23
24
31
32
33
34
41
42
43
44
```

Sample Output

```
11
22
33
44
12
23
34
13
24
14
```

```
#include<iostream>
#include <vector>

using namespace std;

void u_dia_traversal(const vector<vector<int>> &mat){
    int o = mat.size();
    for(int j{};j<o;j++){
        for(int i{};i<o;i++){
            if (i+j<o){
                cout<<mat[i][i+j]<<endl;
            }
        }
    }
}

int main(){
    //order of square matrix
    int o{};
    cin>>o ;

    //filling the matrix
    vector<vector<int>> matrix;
    for(int i{};i<o;i++){
        vector<int> sarr;
        int ele{};
        for(int j{};j<o;j++){
            cin>>ele;
            sarr.push_back(ele);
        }
        matrix.push_back(sarr);
    }

    u_dia_traversal(matrix);

}
```

Saddle Price

Easy

1. You are given a square matrix of size 'n'. You are given n*n elements of the square matrix.
2. You are required to find the saddle price of the given matrix and print the saddle price.
3. The saddle price is defined as the least price in the row but the maximum price in the column of the matrix.

Constraints

```
1 <= n <= 10^2
-10^9 <= e11, e12, .. n * m elements <= 10^9
```

Format

Input

A number n
e11
e12..
e21
e22..
.. n * n number of elements of array a

Output

Saddle point of the matrix if available or "Invalid input" if no saddle point is there.

Example

Sample Input

```
4
11
12
13
14
21
22
23
24
31
32
33
34
41
42
43
```

`44`

Sample Output

`41`

```
#include <iostream>
#include<vector>
using namespace std;
void saddle_point(vector<vector<int>> mat){
    int o = mat.size();






    for(int i{};i<o;i++){
        int a{0};
        int cn{};
        int rmin{mat[i][0]};

        for(int j{};j<o;j++){
            if(mat[i][j]<rmin){
                rmin = mat[i][j];
                cn = j;    //coloum  index of rmin in that row
            }

        }
        for(int k{};k<o;k++){
            if(mat[k][cn]>rmin){
                a = -1;

                break;
            }
        }

        if(a == 0){
            cout<<rmin;
            return ;
        }
    }

    cout<<"Invalid input "<<endl;
    return;
}
int main(){
    //cout<<"Enter the size of 2d array: ";
    int o{};
    cin>>o;


   // cout <<"fill the array";
    vector<vector<int>> mat;
    for(int i{};i<o;i++){
        vector<int> sarr;
        for(int j{};j<o;j++){
            int ele{};
            cin>>ele;
            sarr.push_back(ele);
        }
        mat.push_back(sarr);
    }

    saddle_point(mat);


    return 0;
}
```

Search In A Sorted 2d Array

Medium

1. You are given a number n, representing the number of rows and columns of a square matrix.
2. You are given n * n numbers, representing elements of 2d array a.
Note - Each row and column is sorted in increasing order.
3. You are given a number x.
4. You are required to find x in the matrix and print it's location int (row, col) format as discussed in output format below.
5. In case element is not found, print "Not Found".

Constraints

```
1 <= n <= 10^2
-10^9 <= e11, e12, .. n * m elements <= 10^9
All rows and columns are sorted in increasing order
```

Format

Input

A number n
e11
e12..
e21
e22..
.. n * n number of elements of array a
A number x

Output

row
col of the location where element is found or "Not Found" if element is not in the matrix

Example

Sample Input

```
4
11
12
13
14
21
22
```

```
23
24
31
32
33
34
41
42
43
44
43
```

Sample Output

```
3
2
```

```
#include <iostream>
#include<vector>
using namespace std;
void searh_in_a_2d_sorted_array(vector<vector<int>> mat,int
to_find){
    int o = mat.size();

    int i = 0 ;
    int j = o-1 ;
    while( i<o && j>=0 ){

        if(to_find == mat[i][j]){
            cout<<i<<endl<<j<<endl;
            return;
        }else if(to_find > mat[i][j]){
            i++;
        }else{
            j--;
        }
    }
    cout<<"Not Found"<<endl;

    return;
}
int main(){
    //cout<<"Enter the size of 2d array: ";
    int o{};
    cin>>o;


   // cout <<"fill the array";
    vector<vector<int>> mat;
    for(int i{};i<o;i++){
        vector<int> sarr;

        for(int j{};j<o;j++){
            int ele{};
            cin>>ele;
            sarr.push_back(ele);
        }
        mat.push_back(sarr);
    }
    int to_find{};
    cin >> to_find;

    searh_in_a_2d_sorted_array(mat,to_find);



    return 0;
}
```