Problem Statement – Wave Form Traversal

In the kingdom of **NumMatrix**, the Royal Messenger must deliver letters by walking through the castle halls. The castle is represented as an **N × M matrix**, where each room has a number.

The messenger follows a **wave-like path**:

- In the **first column**, he moves **top to bottom**.
- In the **second column**, he moves **bottom to top**.
- In the **third column**, again **top to bottom**.
- This continues for all columns.

Your task is to print the **wave form traversal** of the matrix.

Input Format

1. The first line contains two integers N and M – the number of rows and columns.
2. The next N × M numbers represent the matrix elements.

Output Format

- Print the elements of the matrix in **wave order traversal** (space-separated).

Constraints

- $1 \leq N, M \leq 100$
- $1 \leq arr[i][j] \leq 10^4$

Example 1

Input

3 4
1 2 3 4
5 6 7 8
9 10 11 12

Output

1 5 9 10 6 2 3 7 11 12 8 4

Example 2

Input

4 3
1 2 3
4 5 6
7 8 9
10 11 12

Output

1 4 7 10 11 8 5 2 3 6 9 12

Code:-

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
   int n;
   int m;
   cin >> n >> m;
   //2D Vector declaration
   vector<vector<int>> v1(n, vector<int>(m,0));
   //Input elements
   for(int i=0;i<n;i++){
      for(int j=0;j<m;j++){
         cin>>v1[i][j];
```

```
    }
  }
  //resultant Vector
  vector<int>v2;
  int fc=0;
  int lc=m-1;
  int fr=0;
  int lr=n-1;
  //top to down;
  while(fc<=lc){
  for(int i=fr;i<=lr;i++){
        v2.push_back(v1[i][fc]);

    }
    fc++;//first column +1
    //down to top
  for(int i=lr;i>=fr;i--){

        v2.push_back(v1[i][fc]);

  }
  fc++;
}

  //Print result
  for(int i=0;i<n*m;i++){
    cout<<v2[i]<<" ";

  }
```

```
    return 0;
}
```

Problem Statement – Transpose of a Matrix

In the **Academy of NumMatrix**, students are given a magical board represented as an **N × M matrix**.
The headmaster wants to flip the matrix along its **main diagonal** so that rows become columns and columns become rows.
This operation is called a **Transpose**.

Your task is to help the headmaster by printing the **transpose of the given matrix**.

Input Format

1. The first line contains two integers N and M – the number of rows and columns.
2. The next N × M numbers represent the matrix elements.

Output Format

- Print the **transpose matrix** (of size M × N).

Constraints

- 1 ≤ N, M ≤ 100
- 1 ≤ arr[i][j] ≤ 10^4

Example 1

Input

```
3 3
1 2 3
4 5 6
7 8 9
```

Output

```
1 4 7
2 5 8
3 6 9
```

Example 2

Input

```
2 3
1 2 3
4 5 6
```

Output

```
1 4
2 5
3 6
```

Code:-

```
 #include <iostream>
#include <vector>
using namespace std;
int main()
{
   int m;
   int n;
   cin >> m >> n;
   //2D Vector declaration
   vector<vector<int>> v1(m, vector<int>(n,0));
   //Input elements
   for(int i=0;i<m;i++){
      for(int j=0;j<n;j++){
         cin>>v1[i][j];
      }
   }
   vector<vector<int>> v2(n, vector<int>(m,0));
   //transpose matrix
   for(int i=0;i<n;i++){
      for(int j=0;j<m;j++){
         v2[i][j]=v1[j][i];

      }
   }
   //print Result
   for(int i=0;i<n;i++){
      for(int j=0;j<m;j++){
         cout<<v2[i][j]<<" ";

      }
      cout<<"\n";
   }

   return 0;
}
```

Problem Statement – Spiral Traversal of a Matrix

In the **Royal Garden of NumMatrix**, the King wants to enjoy the flowers arranged in an **N × M rectangular layout**.
He instructs his gardener to walk in a **spiral path** starting from the **top-left corner**:

1. Walk **left to right** along the top row.
2. Then walk **top to bottom** along the rightmost column.
3. Then walk **right to left** along the bottom row.
4. Then walk **bottom to top** along the leftmost column.
5. Continue the process inward until every element is visited.

Your task is to print the **spiral order traversal** of the given matrix.

Input Format

1. First line: Two integers N and M (rows and columns).
2. Next N × M integers: The elements of the matrix.

Output Format

- Print the matrix elements in **spiral order traversal** (space-separated).

Constraints

- 1 ≤ N, M ≤ 100
- 1 ≤ arr[i][j] ≤ 10^4

Example 1

Input

3 3
1 2 3
4 5 6
7 8 9

Output

1 2 3 6 9 8 7 4 5

Example 2

Input

3 4
1 2 3 4
5 6 7 8
9 10 11 12

Output

1 2 3 4 8 12 11 10 9 5 6 7

Code:

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

using namespace std;
int main()
{
   int m, n;
   cin >> m >> n;
   vector<int> v;
```

```
// 2D vector declaration
vector<vector<int>> arr(m, vector<int>(n));


for (int i = 0; i < m; i++)
{
   for (int j = 0; j < n; j++)
   {
      cin>>arr[i][j] ;
   }
}
int firstRow = 0;
int lastRow = m - 1;
int firstC = 0;
int lastC = n - 1;
while (firstRow <=lastRow && firstC <=lastC)
{
   // Top Row
   for (int j = firstC; j <= lastC; j++)
   {
      v.push_back(arr[firstRow][j]);
   }
   firstRow++;
   // right column
   for (int i = firstRow; i <= lastRow; i++)
   {
      v.push_back(arr[i][lastC]);
   }
   lastC--;
   // bottom Row
```

```
        if(firstRow<=lastRow){
        for (int j = lastC; j >= firstC; j--)
        {
            v.push_back(arr[lastRow][j]);
        }
    }
        lastRow--;
        // left column
        if(firstC<=lastC){
        for (int i = lastRow; i >= firstRow; i--)
        {
            v.push_back(arr[i][firstC]);
        }
    }
        firstC++;
    }
    //Print result

    for (int e : v)
    {
        cout << e << " ";
    }
    return 0;
}
```

Problem Statement – Rotate Matrix by 90° Clockwise

In the **Palace of Geometry**, the Queen wants to **rotate her painting frames** (arranged in an $N \times N$ square matrix).
The Queen commands:

- Rotate the matrix **90 degrees clockwise** (in place).

Your job is to help the palace workers perform this rotation.

Input Format

1. First line: Integer N (size of the square matrix).
2. Next N × N integers: The elements of the matrix.

Output Format

- Print the **rotated matrix** (row by row).

Constraints

- 1 ≤ N ≤ 100
- 1 ≤ arr[i][j] ≤ 10^4

Example 1

Input

```
3
1 2 3
4 5 6
7 8 9
```

Output

```
7 4 1
8 5 2
9 6 3
```

Example 2

Input

```
4
1  2  3  4
5  6  7  8
9 10 11 12
```

13 14 15 16

Output

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

Code:-

```
#include<iostream>
#include<vector>
using namespace std;
int main(){
    int n;
    cin>>n;
    vector<vector<int>> mat(n,vector<int>(n));
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++) {
            cin>>mat[i][j];

        }
    }
    //Transpose Matrix
     for(int i=0;i<n;i++){
        for(int j=i+1;j<n;j++) {
            int temp=mat[i][j];
            mat[i][j]=mat[j][i];
            mat[j][i]=temp;

        }
    }
    //Reverse Rows
    for(int i=0;i<n;i++){
        for(int j=0;j<n/2;j++) {
            int temp=mat[i][j];
            mat[i][j]=mat[i][n-1-j];
            mat[i][n-1-j]=temp;


        }
    }
```

```
    //Print result
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++) {
         cout<< mat[i][j]<<" ";

        }
        cout<<"\n";
    }

    return 0;
}
```