Matrix

1}Given a matrix of size r*c. Traverse the matrix in spiral form.

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

int main() {
    int r, c;
    cout << "Enter number of rows: ";
    cin >> r;
    cout << "Enter number of columns: ";
    cin >> c;

    vector<vector<int>> matrix(r, vector<int>(c));

    cout << "Enter the elements of the matrix:" << endl;
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            cin >> matrix[i][j];
        }
    }

    cout << "Spiral traversal of matrix: ";

    int top = 0, bottom = r - 1, left = 0, right = c - 1;

    while (top <= bottom && left <= right) {
        // 1️⃣ Traverse from left → right
```

```
    while (top <= bottom && left <= right) {
        // 1️⃣ Traverse from left → right
        for (int i = left; i <= right; i++)
            cout << matrix[top][i] << " ";
        top++;

        // 2️⃣ Traverse from top → bottom
        for (int i = top; i <= bottom; i++)
            cout << matrix[i][right] << " ";
        right--;

        // 3️⃣ Traverse from right → left (if still within bounds)
        if (top <= bottom) {
            for (int i = right; i >= left; i--)
                cout << matrix[bottom][i] << " ";
            bottom--;
        }

        // 4️⃣ Traverse from bottom → top (if still within bounds)
        if (left <= right) {
            for (int i = bottom; i >= top; i--)
                cout << matrix[i][left] << " ";
            left++;
        }
    }

```

Output:

```
Enter number of rows: 2
Enter number of columns: 2
Enter the elements of the matrix:
12
13
14
15
Spiral traversal of matrix: 12 13 15 14


=== Code Execution Successful ===
```

2}Given a binary matrix M of size n X m. Find the maximum area of a rectangle formed only of 1s in the given matrix.

```cpp
#include <iostream>
#include <vector>
#include <stack>
#include <algorithm>
using namespace std;

// Function to find largest rectangle area in a histogram
int largestRectangleArea(vector<int>& heights) {
    stack<int> s;
    int maxArea = 0;
    heights.push_back(0); // Sentinel value to flush remaining bars

    for (int i = 0; i < heights.size(); i++) {
        while (!s.empty() && heights[s.top()] > heights[i]) {
            int height = heights[s.top()];
            s.pop();
            int width = s.empty() ? i : i - s.top() - 1;
            maxArea = max(maxArea, height * width);
        }
        s.push(i);
    }
    heights.pop_back();
    return maxArea;
}
```

```cpp
int maxRectangle(vector<vector<int>>& M) {
    if (M.empty()) return 0;

    int n = M.size(), m = M[0].size();
    vector<int> height(m, 0);
    int maxArea = 0;

    for (int i = 0; i < n; i++) {
        // Update height array for current row
        for (int j = 0; j < m; j++) {
            height[j] = (M[i][j] == 0) ? 0 : height[j] + 1;
        }

        // Find largest rectangle area for current histogram
        maxArea = max(maxArea, largestRectangleArea(height));
    }
    return maxArea;
}

int main() {
    int n, m;
    cout << "Enter number of rows: ";
    cin >> n;
    cout << "Enter number of columns: ";
    cin >> m;
```

```cpp
        }
    }

    cout << "Maximum area of rectangle of 1s: " << maxRectangle(M)
        << endl;
    return 0;
}
```

Output:

```
Enter number of rows: 2
Enter number of columns: 2
Enter binary matrix (0s and 1s):
12
11
14
15
Maximum area of rectangle of 1s: 4


=== Code Execution Successful ===
```

2}Given a square matrix, turn it by 90 degrees in a clockwise direction without using any extra space.

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

#define N 3 // You can change the size of the square matrix here

// Function to rotate the matrix by 90 degrees clockwise
void rotate90Clockwise(int matrix[N][N]) {
    // Step 1: Transpose the matrix
    for (int i = 0; i < N; i++) {
        for (int j = i; j < N; j++) {
            swap(matrix[i][j], matrix[j][i]);
        }
    }

    // Step 2: Reverse each row
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N / 2; j++) {
            swap(matrix[i][j], matrix[i][N - j - 1]);
        }
    }
}

// Function to print the matrix
void printMatrix(int matrix[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
```

```cpp
// Function to print the matrix
void printMatrix(int matrix[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            cout << matrix[i][j] << " ";
        cout << endl;
    }
}

int main() {
    int matrix[N][N] = { {1, 2, 3},
                         {4, 5, 6},
                         {7, 8, 9} };

    cout << "Original Matrix:\n";
    printMatrix(matrix);

    rotate90Clockwise(matrix);

    cout << "\nMatrix after 90-degree clockwise rotation:\n";
    printMatrix(matrix);

    return 0;
}
```

Output:

```
Original Matrix:
1 2 3
4 5 6
7 8 9

Matrix after 90-degree clockwise rotation:
7 4 1
8 5 2
9 6 3
```

3} Take a matrix and checks if it is invertible. An invertible matrix is a square matrix whose determinant exists.

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

// Function to find the determinant of a matrix (recursive method)
int determinant(int matrix[10][10], int n) {
    int det = 0;
    int submatrix[10][10];

    if (n == 1)
        return matrix[0][0];

    if (n == 2)
        return (matrix[0][0] * matrix[1][1]) - (matrix[0][1] *
            matrix[1][0]);

    for (int x = 0; x < n; x++) {
        int subi = 0;
        for (int i = 1; i < n; i++) {
            int subj = 0;
            for (int j = 0; j < n; j++) {
                if (j == x)
                    continue;
                submatrix[subi][subj] = matrix[i][j];
                subj++;
            }
            subi++;
```

```
            subi++;
        }
        det += (x % 2 == 0 ? 1 : -1) * matrix[0][x] * determinant
            (submatrix, n - 1);
    }
    return det;
}

int main() {
    int n;
    int matrix[10][10];

    cout << "Enter the order of the square matrix: ";
    cin >> n;

    cout << "Enter elements of the matrix:\n";
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> matrix[i][j];
        }
    }

    cout << "\nMatrix entered:\n";
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            cout << matrix[i][j] << " ";
```

```
    cout << "\nMatrix entered:\n";
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            cout << matrix[i][j] << " ";
        cout << endl;
    }

    int det = determinant(matrix, n);
    cout << "\nDeterminant of the matrix = " << det << endl;

    if (det != 0)
        cout << "✅ The matrix is invertible (non-singular)." << endl
            ;
    else
        cout << "❌ The matrix is NOT invertible (singular)." << endl
            ;

    return 0;
}
```

Output:

```
Enter the order of the square matrix: 2
Enter elements of the matrix:
23
23
34
45

Matrix entered:
23 23
34 45

Determinant of the matrix = 253
✅ The matrix is invertible (non-singular).
```