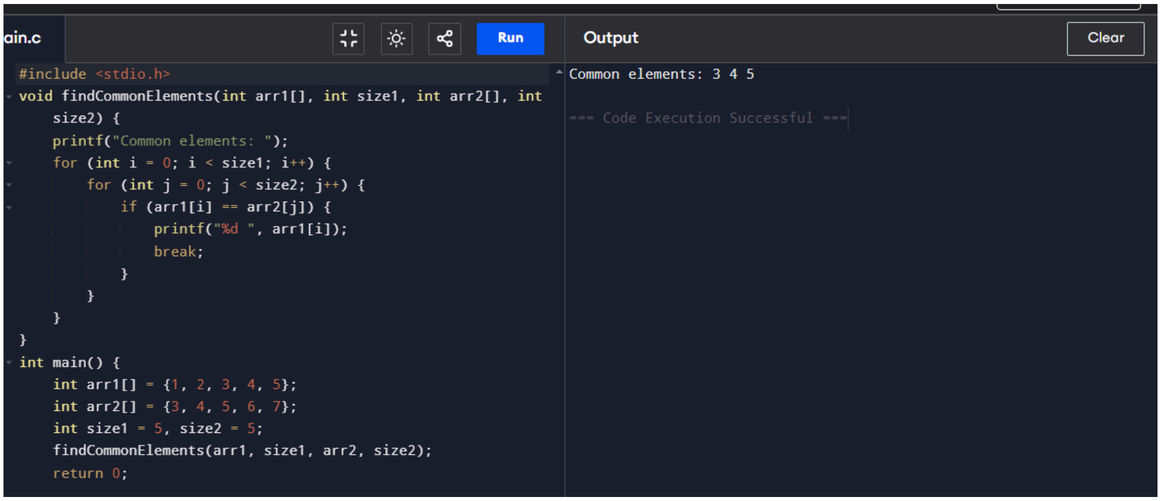
1. Given two arrays of integers, find the common elements between them.

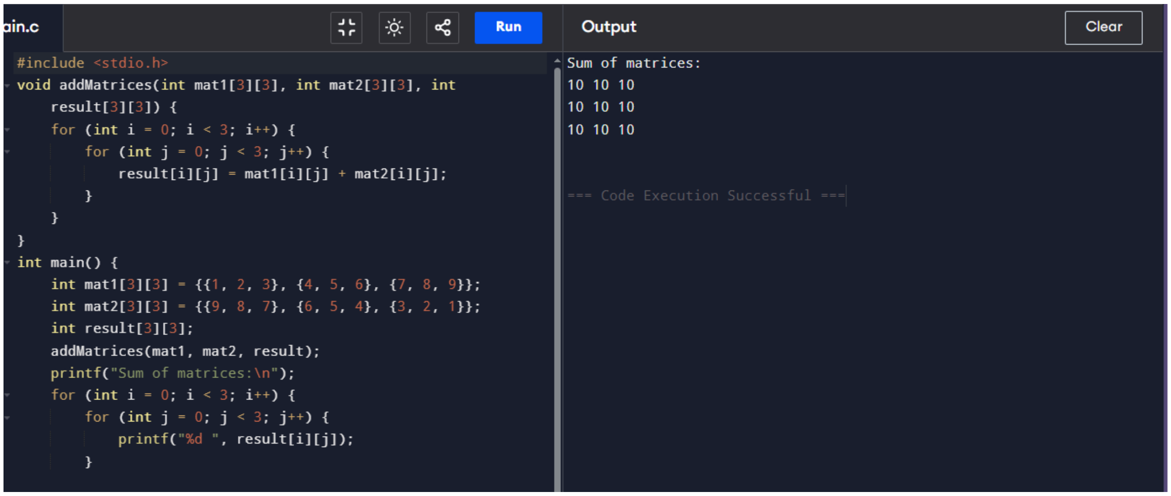


```c
#include <stdio.h>
void findCommonElements(int arr1[], int size1, int arr2[], int
    size2) {
    printf("Common elements: ");
    for (int i = 0; i < size1; i++) {
        for (int j = 0; j < size2; j++) {
            if (arr1[i] == arr2[j]) {
                printf("%d ", arr1[i]);
                break;
            }
        }
    }
}
int main() {
    int arr1[] = {1, 2, 3, 4, 5};
    int arr2[] = {3, 4, 5, 6, 7};
    int size1 = 5, size2 = 5;
    findCommonElements(arr1, size1, arr2, size2);
    return 0;
```

```
Common elements: 3 4 5

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

2. Write a program to add two 3x3 matrices using a two-dimensional array.

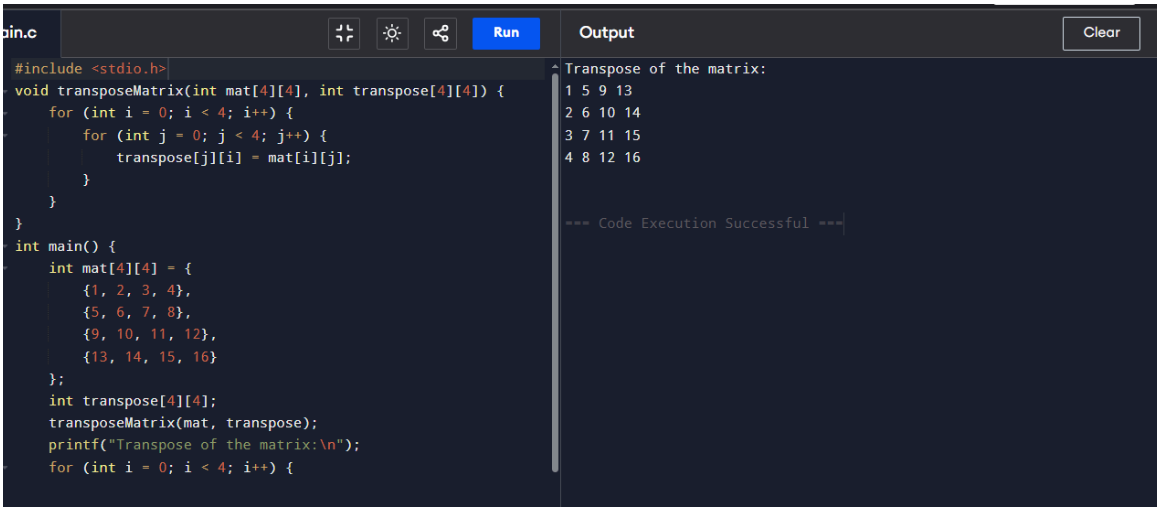
```c
#include <stdio.h>

void addMatrices(int mat1[3][3], int mat2[3][3], int result[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            result[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
}

int main() {
    int mat1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int mat2[3][3] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
    int result[3][3];

    addMatrices(mat1, mat2, result);

    printf("Sum of matrices:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```



```
#include <stdio.h>
void addMatrices(int mat1[3][3], int mat2[3][3], int
    result[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            result[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
}
int main() {
    int mat1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int mat2[3][3] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
    int result[3][3];
    addMatrices(mat1, mat2, result);
    printf("Sum of matrices:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", result[i][j]);
        }
```

```
Sum of matrices:
10 10 10
10 10 10
10 10 10


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

3. Write a program to compute the transpose of a 4x4 matrix using two-dimensional arrays.

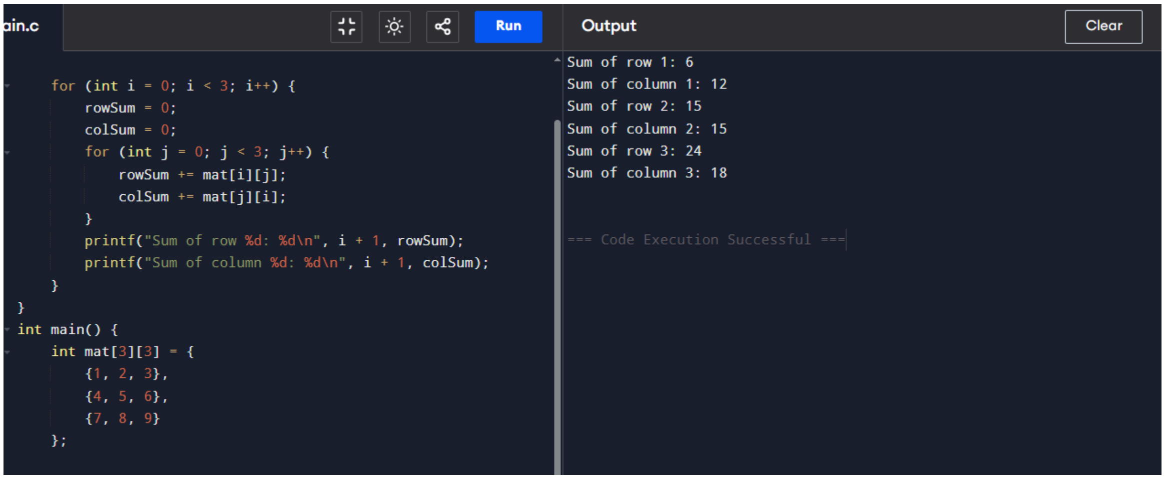


```
#include <stdio.h>
void transposeMatrix(int mat[4][4], int transpose[4][4]) {
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            transpose[j][i] = mat[i][j];
        }
    }
}
int main() {
    int mat[4][4] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12},
        {13, 14, 15, 16}
    };
    int transpose[4][4];
    transposeMatrix(mat, transpose);
    printf("Transpose of the matrix:\n");
    for (int i = 0; i < 4; i++) {
```

```
Transpose of the matrix:
1 5 9 13
2 6 10 14
3 7 11 15
4 8 12 16


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

```
#include <stdio.h>

void transposeMatrix(int mat[4][4], int transpose[4][4]) {
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            transpose[j][i] = mat[i][j];
        }
    }
}

int main() {
    int mat[4][4] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12},
        {13, 14, 15, 16}
    };
    int transpose[4][4];
    transposeMatrix(mat, transpose);

    printf("Transpose of the matrix:\n");
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            printf("%d ", transpose[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

4. Write a program to compute and display the sum of rows and columns of a 3x3 matrix.



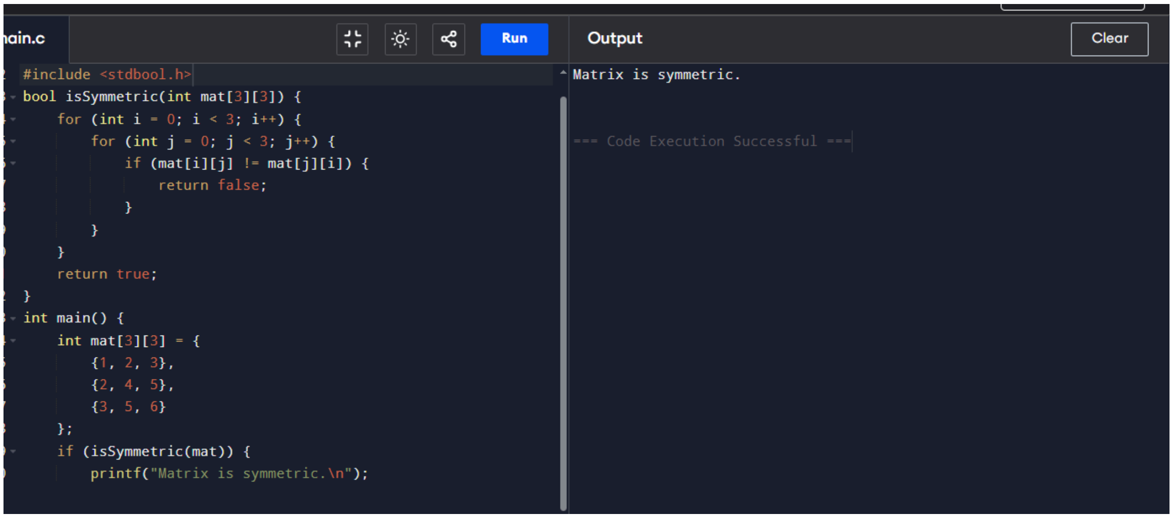
```
    for (int i = 0; i < 3; i++) {
        rowSum = 0;
        colSum = 0;
        for (int j = 0; j < 3; j++) {
            rowSum += mat[i][j];
            colSum += mat[j][i];
        }
        printf("Sum of row %d: %d\n", i + 1, rowSum);
        printf("Sum of column %d: %d\n", i + 1, colSum);
    }
}
int main() {
    int mat[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };
```

```
Sum of row 1: 6
Sum of column 1: 12
Sum of row 2: 15
Sum of column 2: 15
Sum of row 3: 24
Sum of column 3: 18


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

```
#include <stdio.h>

void sumRowsAndColumns(int mat[3][3]) {
    int rowSum, colSum;

    for (int i = 0; i < 3; i++) {
        rowSum = 0;
        colSum = 0;
        for (int j = 0; j < 3; j++) {
            rowSum += mat[i][j];
            colSum += mat[j][i];
        }
        printf("Sum of row %d: %d\n", i + 1, rowSum);
        printf("Sum of column %d: %d\n", i + 1, colSum);
    }
}

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

    sumRowsAndColumns(mat);
    return 0;
}
```

5. Write a program to determine if a given 3x3 matrix is a symmetric matrix.

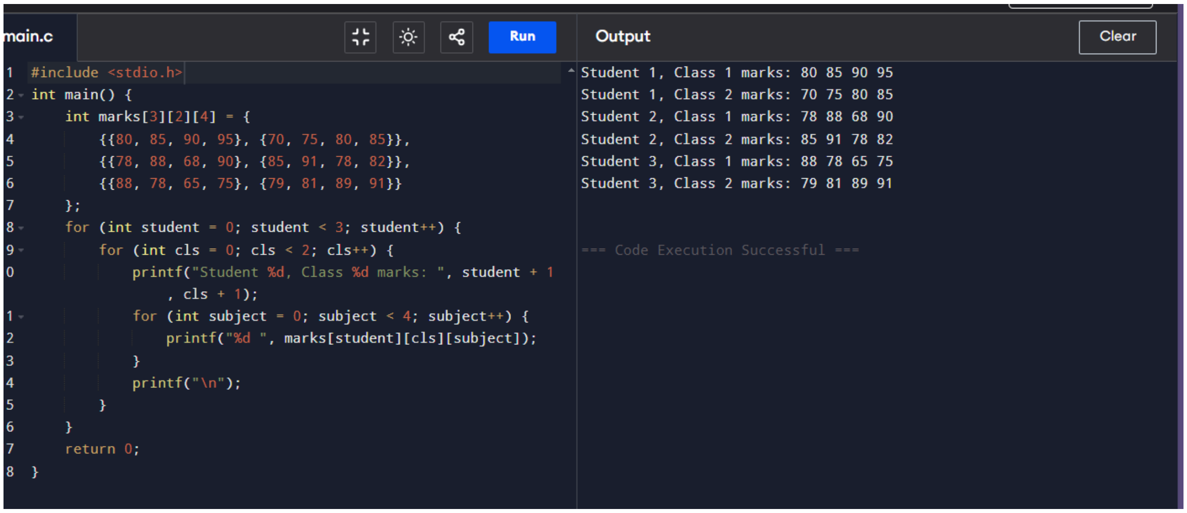
```
#include <stdio.h>
#include <stdbool.h>

bool isSymmetric(int mat[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (mat[i][j] != mat[j][i]) {
                return false;
            }
        }
    }
    return true;
}

int main() {
    int mat[3][3] = {
        {1, 2, 3},
        {2, 4, 5},
        {3, 5, 6}
    };

    if (isSymmetric(mat)) {
        printf("Matrix is symmetric.\n");
    } else {
        printf("Matrix is not symmetric.\n");
    }
    return 0;
}
```



```c
#include <stdbool.h>
bool isSymmetric(int mat[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (mat[i][j] != mat[j][i]) {
                return false;
            }
        }
    }
    return true;
}
int main() {
    int mat[3][3] = {
        {1, 2, 3},
        {2, 4, 5},
        {3, 5, 6}
    };
    if (isSymmetric(mat)) {
        printf("Matrix is symmetric.\n");
```

```
Matrix is symmetric.


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

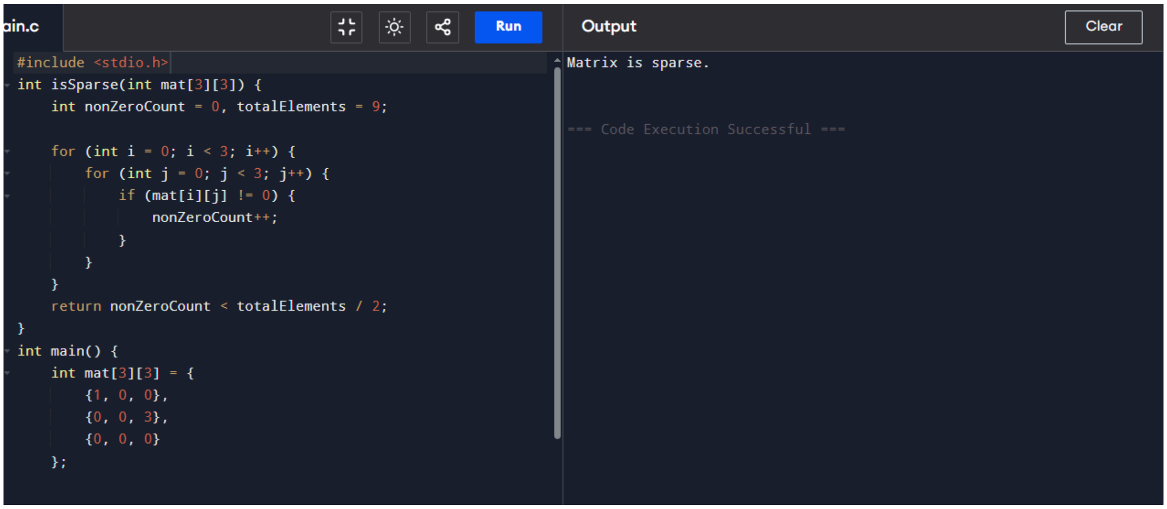
6. Write a program to use a three-dimensional array to store and display the marks of 3 students in 4 subjects for 2 classes.



```c
#include <stdio.h>
int main() {
    int marks[3][2][4] = {
        {{80, 85, 90, 95}, {70, 75, 80, 85}},
        {{78, 88, 68, 90}, {85, 91, 78, 82}},
        {{88, 78, 65, 75}, {79, 81, 89, 91}}
    };
    for (int student = 0; student < 3; student++) {
        for (int cls = 0; cls < 2; cls++) {
            printf("Student %d, Class %d marks: ", student + 1
                , cls + 1);
            for (int subject = 0; subject < 4; subject++) {
                printf("%d ", marks[student][cls][subject]);
            }
            printf("\n");
        }
    }
    return 0;
}
```

```
Student 1, Class 1 marks: 80 85 90 95
Student 1, Class 2 marks: 70 75 80 85
Student 2, Class 1 marks: 78 88 68 90
Student 2, Class 2 marks: 85 91 78 82
Student 3, Class 1 marks: 88 78 65 75
Student 3, Class 2 marks: 79 81 89 91


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

7. Write a program to check whether a matrix is sparse and display its non-zero elements



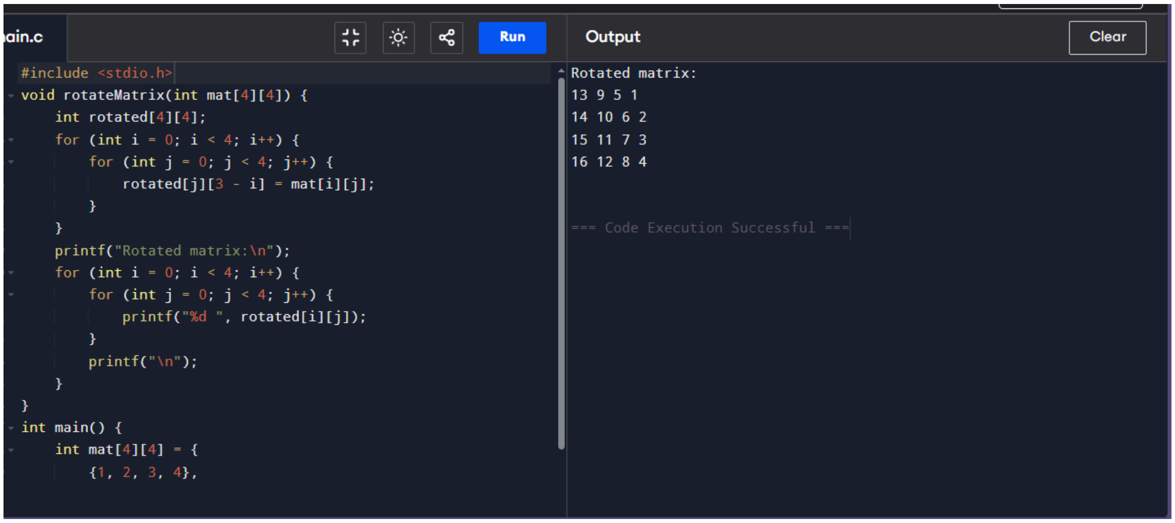
```c
#include <stdio.h>
int isSparse(int mat[3][3]) {
    int nonZeroCount = 0, totalElements = 9;

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (mat[i][j] != 0) {
                nonZeroCount++;
            }
        }
    }
    return nonZeroCount < totalElements / 2;
}
int main() {
    int mat[3][3] = {
        {1, 0, 0},
        {0, 0, 3},
        {0, 0, 0}
    };
```

```
Matrix is sparse.


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

#include <stdio.h>

```
int isSparse(int mat[3][3]) {
    int nonZeroCount = 0, totalElements = 9;

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (mat[i][j] != 0) {
                nonZeroCount++;
            }
        }
    }
    return nonZeroCount < totalElements / 2;
}

int main() {
    int mat[3][3] = {
        {1, 0, 0},
        {0, 0, 3},
        {0, 0, 0}
    };

    if (isSparse(mat)) {
        printf("Matrix is sparse.\n");
    } else {
        printf("Matrix is not sparse.\n");
    }
    return 0;
}
```

8. Write a program to rotate a 4x4 matrix 90 degrees in the clockwise direction.

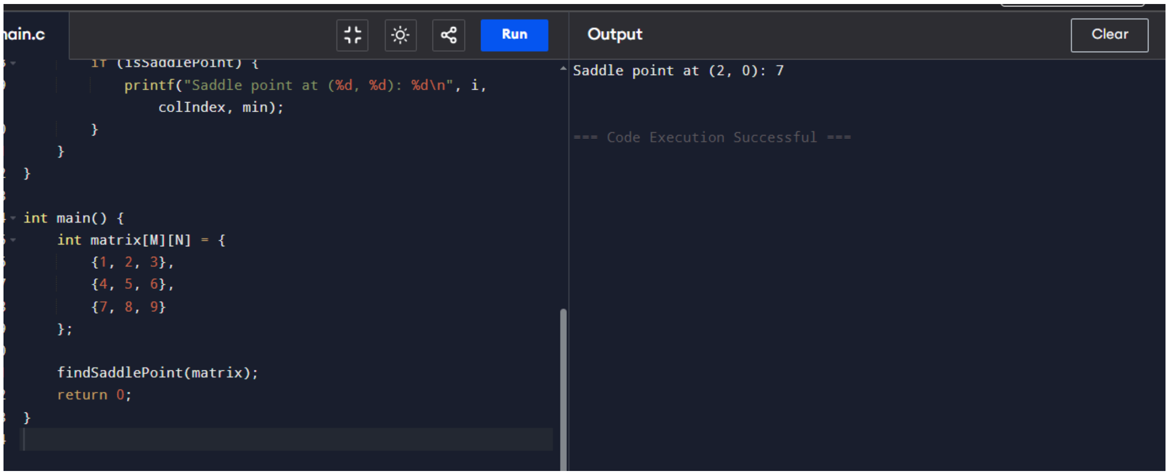


```
#include <stdio.h>
void rotateMatrix(int mat[4][4]) {
    int rotated[4][4];
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            rotated[j][3 - i] = mat[i][j];
        }
    }
    printf("Rotated matrix:\n");
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            printf("%d ", rotated[i][j]);
        }
        printf("\n");
    }
}
int main() {
    int mat[4][4] = {
        {1, 2, 3, 4},
```

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


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

```
#include <stdio.h>

void rotateMatrix(int mat[4][4]) {
    int rotated[4][4];
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            rotated[j][3 - i] = mat[i][j];
        }
    }

    printf("Rotated matrix:\n");
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            printf("%d ", rotated[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int mat[4][4] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12},
        {13, 14, 15, 16}
    };

    rotateMatrix(mat);
    return 0;
}
```

9. Given a matrix (2D array) of integers, find the saddle point(s) (an element that is the minimum in its row and maximum in its column)

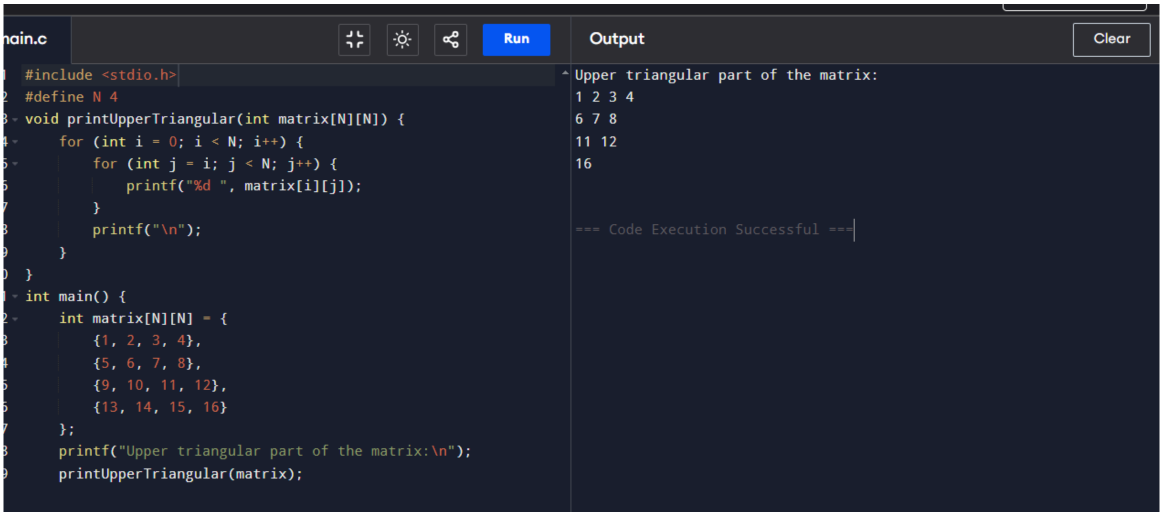


```
        if (isSaddlePoint) {
            printf("Saddle point at (%d, %d): %d\n", i,
                colIndex, min);
        }
    }
}

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

    findSaddlePoint(matrix);
    return 0;
}
```

```
Saddle point at (2, 0): 7


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

```
#include <stdio.h>

#define M 3  // Number of rows
#define N 3  // Number of columns

void findSaddlePoint(int matrix[M][N]) {
    for (int i = 0; i < M; i++) {
        int min = matrix[i][0], colIndex = 0;
        // Find the minimum element in the row
        for (int j = 1; j < N; j++) {
            if (matrix[i][j] < min) {
                min = matrix[i][j];
                colIndex = j;
            }
        }

        // Check if this element is the maximum in its column
        int isSaddlePoint = 1;
        for (int k = 0; k < M; k++) {
            if (matrix[k][colIndex] > min) {
                isSaddlePoint = 0;
                break;
            }
        }

        // Print the saddle point if found
        if (isSaddlePoint) {
            printf("Saddle point at (%d, %d): %d\n", i, colIndex, min);
        }
    }
}

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

    findSaddlePoint(matrix);
    return 0;
}
```

10. Write a program to print the upper triangular part of a 4x4 matrix.



```c
#include <stdio.h>
#define N 4
void printUpperTriangular(int matrix[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = i; j < N; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}
int main() {
    int matrix[N][N] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12},
        {13, 14, 15, 16}
    };
    printf("Upper triangular part of the matrix:\n");
    printUpperTriangular(matrix);
```

```
Upper triangular part of the matrix:
1 2 3 4
6 7 8
11 12
16


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

```c
#include <stdio.h>

#define N 4  // Matrix size (4x4)

void printUpperTriangular(int matrix[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = i; j < N; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int matrix[N][N] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12},
        {13, 14, 15, 16}
    };

    printf("Upper triangular part of the matrix:\n");
    printUpperTriangular(matrix);
    return 0;
}
```