

## Assessment 1:

Q. Write a program for swapping rows and columns in a 2D matrix to include validation, edge case handling, and multiple operations in one execution. The program should allow users to dynamically modify a matrix based on their input and perform multiple swaps efficiently.

Detailed Requirements:

Matrix Initialization:

Dynamically allocate a 2D matrix of size  $N \times N$ , where  $N$  is provided by the user at runtime.

Populate the matrix with either user input values or randomly generated integers between 1 and 100.

Input Constraints:

Validate that  $N \geq 2$  (to allow swapping operations).

Validate user inputs for row and column indices to ensure they are within valid bounds ( $0 \leq \text{index} < N$ ).

Core Functionality:

Allow the user to swap:

Two rows: Specify the indices of the rows to be swapped.

Two columns: Specify the indices of the columns to be swapped.

Perform both row and column swaps in a single operation (if requested).

Display the matrix after each swap.

Additional Features:

Allow users to perform multiple swaps in a single execution until they choose to exit.

Allow swapping of the same row/column with itself (resulting in no change but clearly indicate this in the output).

Edge Case Handling:

Prevent invalid operations such as attempting to swap rows/columns outside the valid range.

Handle scenarios where the matrix is symmetric, diagonal, or sparse, ensuring correct behavior.

Optional Features:

Allow users to reset the matrix to its original state at any point.

Provide the option to undo the last swap operation.

Performance Optimization:

Use efficient algorithms for in-place swapping without additional memory allocation.

Optimize the matrix display for larger sizes (e.g., truncate large matrices with ellipses in the middle).

Output Requirements:

Display the initial matrix in a clear, structured format.

For each swap, clearly indicate:

Rows and/or columns being swapped.

The updated matrix.

Indicate when no changes occurred due to invalid or redundant swaps.

Example Execution:

Input:

Matrix size  $N=4N = 4N=4$ .

Initial matrix:

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

Swap operations:

Swap row 1 with row 3.

Swap column 0 with column 2.

Processing:

Row Swap:

Swap row 1 with row 3:

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

Column Swap:

Swap column 0 with column 2:

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

Output:

Initial Matrix:

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

After swapping row 1 with row 3:

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

After swapping column 0 with column 2:

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

```

#include <stdio.h>
// Function to get the size of the matrix from the user
int size(){
    int N;
    do{
        printf("Enter the size of the matrix (greater than 2):\n");
        scanf("%d", &N);
        if (N <=2) {
            printf("Size should be larger than 2\n");
        }
    } while (N <= 2);
    return N;
}

```

```

//Display Matrix
void displayMatrix(int matrix[][100],int N){
    for (int i =0; i<N; i++) {
        for (int j =0; j <N; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

```

```

#include <stdio.h>
// Function to get the size of the matrix from the user
int size(){
    int N;
    do{
        printf("Enter the size of the matrix (greater than 2):\n");
        scanf("%d", &N);
        if (N <=2) {
            printf("Size should be larger than 2\n");
        }
    } while (N <= 2);
    return N;
}

```

```

//Display Matrix
void displayMatrix(int matrix[][100],int N){
    for (int i =0; i<N; i++) {
        for (int j =0; j <N; j++) {
            printf("%d ", matrix[i][j]);
        }
    }
}

```

```

    }
    printf("\n");
}
}
int main(){
    int N=size();
    int matrix[100][100];

    // User input for matrix values
    printf("Enter the values to be inserted into the matrix:\n");
    for (int i =0; i<N; i++) {
        for (int j = 0; j < N; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }

    //storing initial matrix
    int initialMatrix[100][100];
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            initialMatrix[i][j] = matrix[i][j];
        }
    }

    displayMatrix(matrix,N);

    // Menu
    int choice;
    do {
        printf("Menu:\n");
        printf("1. Column Swap\n");
        printf("2. Row Swap\n");
        printf("3. Display\n");
        printf("4. Reset Matrix\n");
        printf("5. Exit\n");
        printf("Enter your choice:\n");
        scanf("%d", &choice);

        switch (choice) {
            case 1:{
                // Swapping columns
                int colSwap1, colSwap2;
                printf("Initial Matrix:\n");

```

```

displayMatrix(initialMatrix,N);
printf("Specify the indices of the columns to be swapped (0 to %d):\n", N - 1);
scanf("%d %d", &colSwap1, &colSwap2);
if (colSwap1 >= N || colSwap2 >= N || colSwap1 < 0 || colSwap2 < 0) {
    printf("Enter valid column indices\n");
} else {
    for (int i = 0; i < N; i++) {
        int temp = matrix[i][colSwap1];
        matrix[i][colSwap1] = matrix[i][colSwap2];
        matrix[i][colSwap2] = temp;
    }
    printf("After swapping columns:\n");
    displayMatrix(matrix, N);
}
break;
}
case 2:{
    // Swapping rows
    int rowSwap1, rowSwap2;
    printf("Initial Matrix:\n");
    displayMatrix(initialMatrix,N);
    printf("Specify the indices of the rows to be swapped (0 to %d):\n", N - 1);
    scanf("%d %d", &rowSwap1, &rowSwap2);
    if (rowSwap1 >= N || rowSwap2 >= N || rowSwap1 < 0 || rowSwap2 < 0) {
        printf("Enter valid row indices\n");
    } else {
        for (int j = 0; j < N; j++) {
            int temp = matrix[rowSwap1][j];
            matrix[rowSwap1][j] = matrix[rowSwap2][j];
            matrix[rowSwap2][j] = temp;
        }
        printf("After swapping rows:\n");
        displayMatrix(matrix, N);
    }
    break;
}
case 3:{
    printf("Initial Matrix:\n");
    displayMatrix(initialMatrix,N);
    printf("Current Matrix\n");
    displayMatrix(matrix,N);
    break;
}
case 4:{

```

```

        for (int i=0; i<N; i++){
            for (int j=0; j<N; j++) {
                matrix[i][j] = initialMatrix[i][j];
            }
        }
        break;
    }
    case 5:
        printf("Exiting...\n");
        break;
    default:
        printf("Invalid choice. Please try again.\n");
    }
} while(choice !=5);

return 0;
}

}
}
int main(){
    int N=size();
    int matrix[100][100];

    // User input for matrix values
    printf("Enter the values to be inserted into the matrix:\n");
    for (int i =0; i<N; i++) {
        for (int j = 0; j < N; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }

    // Display matrix

    displayMatrix(matrix,N);

    // Menu
    int choice;
    do {
        printf("Menu:\n");
        printf("1. Column Swap\n");
        printf("2. Row Swap\n");
        printf("3. Display\n");
        printf("4. Exit\n");

```

```

printf("Enter your choice:\n");
scanf("%d", &choice);

switch (choice) {
    case 1:{
        // Swapping columns
        int colSwap1, colSwap2;
        printf("Specify the indices of the columns to be swapped (0 to %d):\n", N - 1);
        scanf("%d %d", &colSwap1, &colSwap2);
        if (colSwap1 >= N || colSwap2 >= N || colSwap1 < 0 || colSwap2 < 0) {
            printf("Enter valid column indices\n");
        } else {
            for (int i = 0; i < N; i++) {
                int temp = matrix[i][colSwap1];
                matrix[i][colSwap1] = matrix[i][colSwap2];
                matrix[i][colSwap2] = temp;
            }
            printf("After swapping columns:\n");
            displayMatrix(matrix, N);
        }
        break;
    }
    case 2:{
        // Swapping rows
        int rowSwap1, rowSwap2;
        printf("Specify the indices of the rows to be swapped (0 to %d):\n", N - 1);
        scanf("%d %d", &rowSwap1, &rowSwap2);
        if (rowSwap1 >= N || rowSwap2 >= N || rowSwap1 < 0 || rowSwap2 < 0) {
            printf("Enter valid row indices\n");
        } else {
            for (int j = 0; j < N; j++) {
                int temp = matrix[rowSwap1][j];
                matrix[rowSwap1][j] = matrix[rowSwap2][j];
                matrix[rowSwap2][j] = temp;
            }
            printf("After swapping rows:\n");
            displayMatrix(matrix, N);
        }
        break;
    }
    case 3:{
        displayMatrix(matrix, N);
        break;
    }
}

```

```
        case 4:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice. Please try again.\n");
    }
} while(choice !=4);

return 0;
}
```