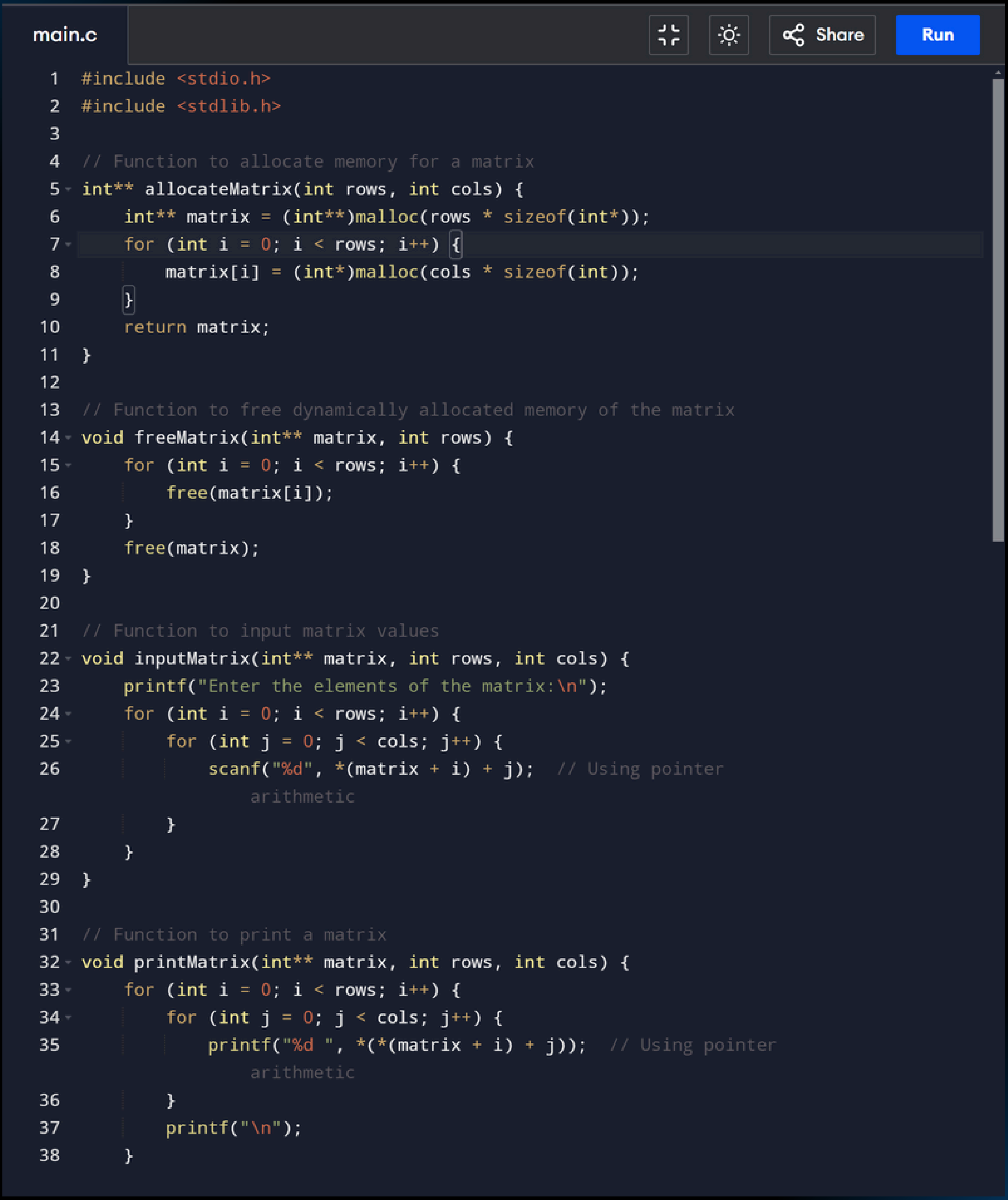
DSA

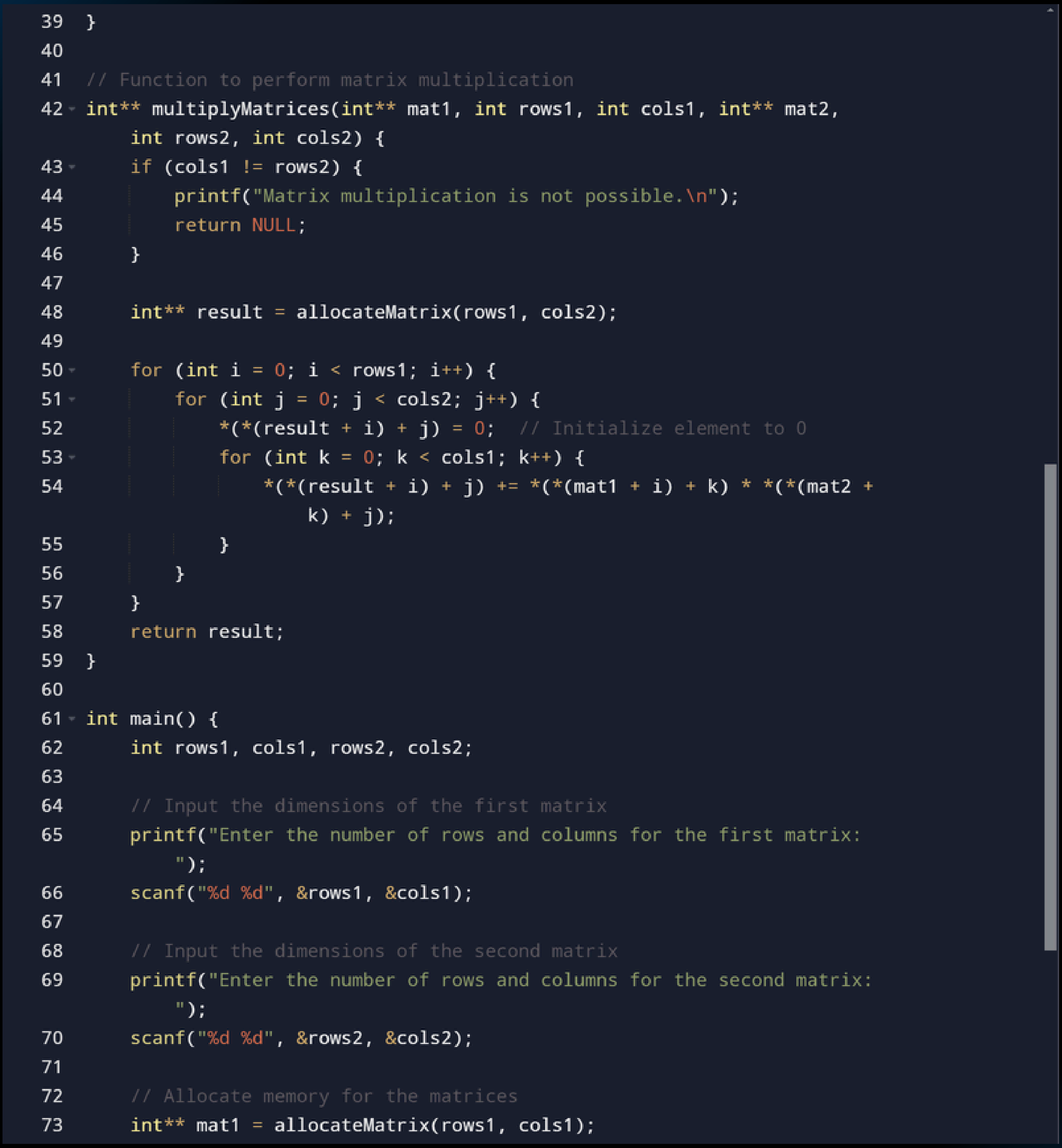
ASSIGMENT-2

Presented By
ADHITHIYAN.B
RA2311026050030
GITHUB:HTTPS://GITHUB.COM/ADHI0007

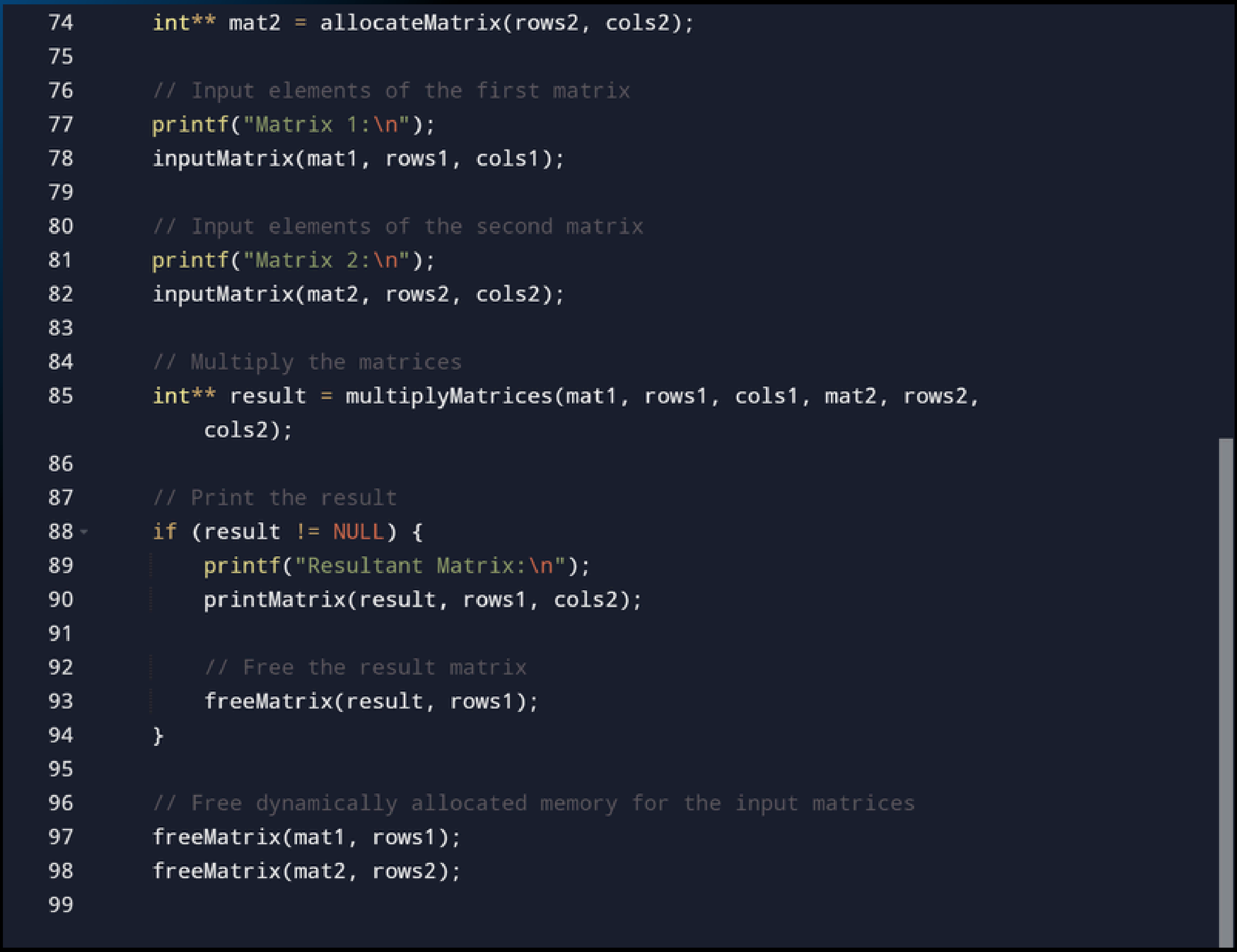
1. Implementation of Matrix Multiplication using Dynamic Memory Allocation. Ensure to allocate the memory using appropriate functions and access the array using pointers.



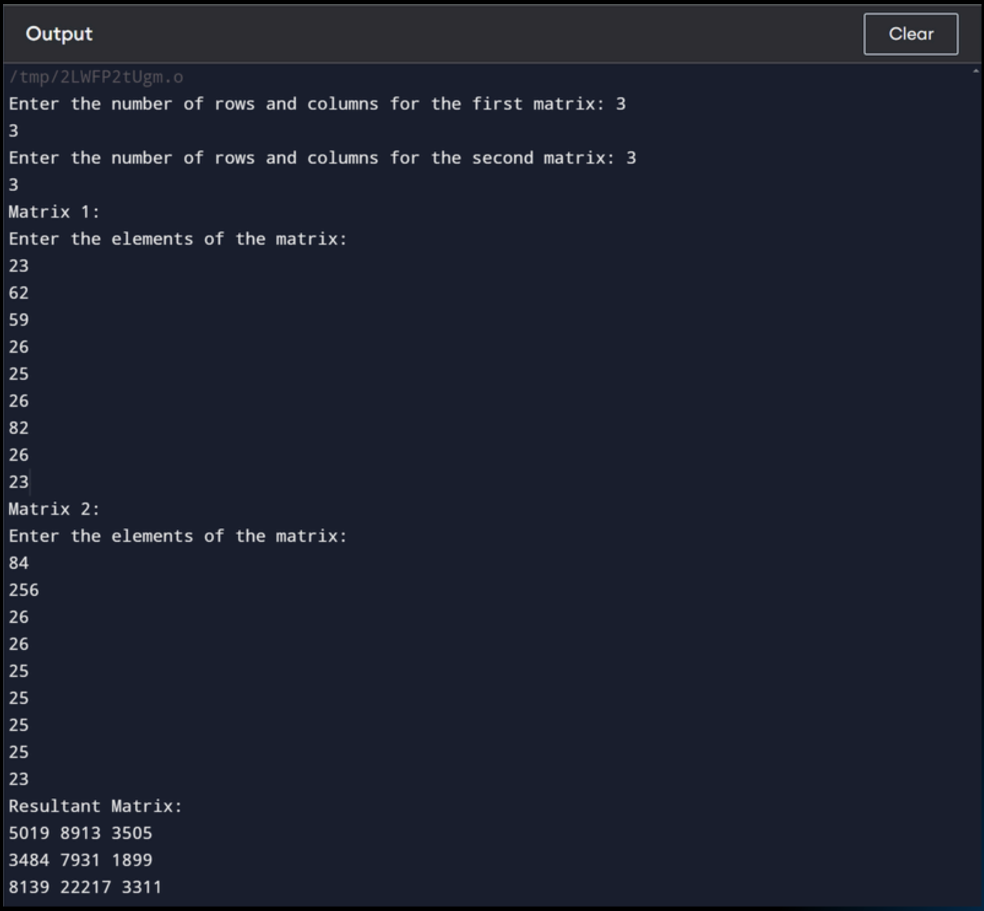
```c
#include <stdio.h>
#include <stdlib.h>

// Function to allocate memory for a matrix
int** allocateMatrix(int rows, int cols) {
    int** matrix = (int**)malloc(rows * sizeof(int*));
    for (int i = 0; i < rows; i++) {
        matrix[i] = (int*)malloc(cols * sizeof(int));
    }
    return matrix;
}

// Function to free dynamically allocated memory of the matrix
void freeMatrix(int** matrix, int rows) {
    for (int i = 0; i < rows; i++) {
        free(matrix[i]);
    }
    free(matrix);
}

// Function to input matrix values
void inputMatrix(int** matrix, int rows, int cols) {
    printf("Enter the elements of the matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", *(matrix + i) + j);  // Using pointer
                arithmetic
        }
    }
}

// Function to print a matrix
void printMatrix(int** matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", *(*(matrix + i) + j));  // Using pointer
                arithmetic
        }
        printf("\n");
    }
```



```c
}

// Function to perform matrix multiplication
int** multiplyMatrices(int** mat1, int rows1, int cols1, int** mat2, int rows2, int cols2) {
    if (cols1 != rows2) {
        printf("Matrix multiplication is not possible.\n");
        return NULL;
    }

    int** result = allocateMatrix(rows1, cols2);

    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols2; j++) {
            *(*(result + i) + j) = 0;  // Initialize element to 0
            for (int k = 0; k < cols1; k++) {
                *(*(result + i) + j) += *(*(mat1 + i) + k) * *(*(mat2 + k) + j);
            }
        }
    }
    return result;
}

int main() {
    int rows1, cols1, rows2, cols2;

    // Input the dimensions of the first matrix
    printf("Enter the number of rows and columns for the first matrix: ");
    scanf("%d %d", &rows1, &cols1);

    // Input the dimensions of the second matrix
    printf("Enter the number of rows and columns for the second matrix: ");
    scanf("%d %d", &rows2, &cols2);

    // Allocate memory for the matrices
    int** mat1 = allocateMatrix(rows1, cols1);
```



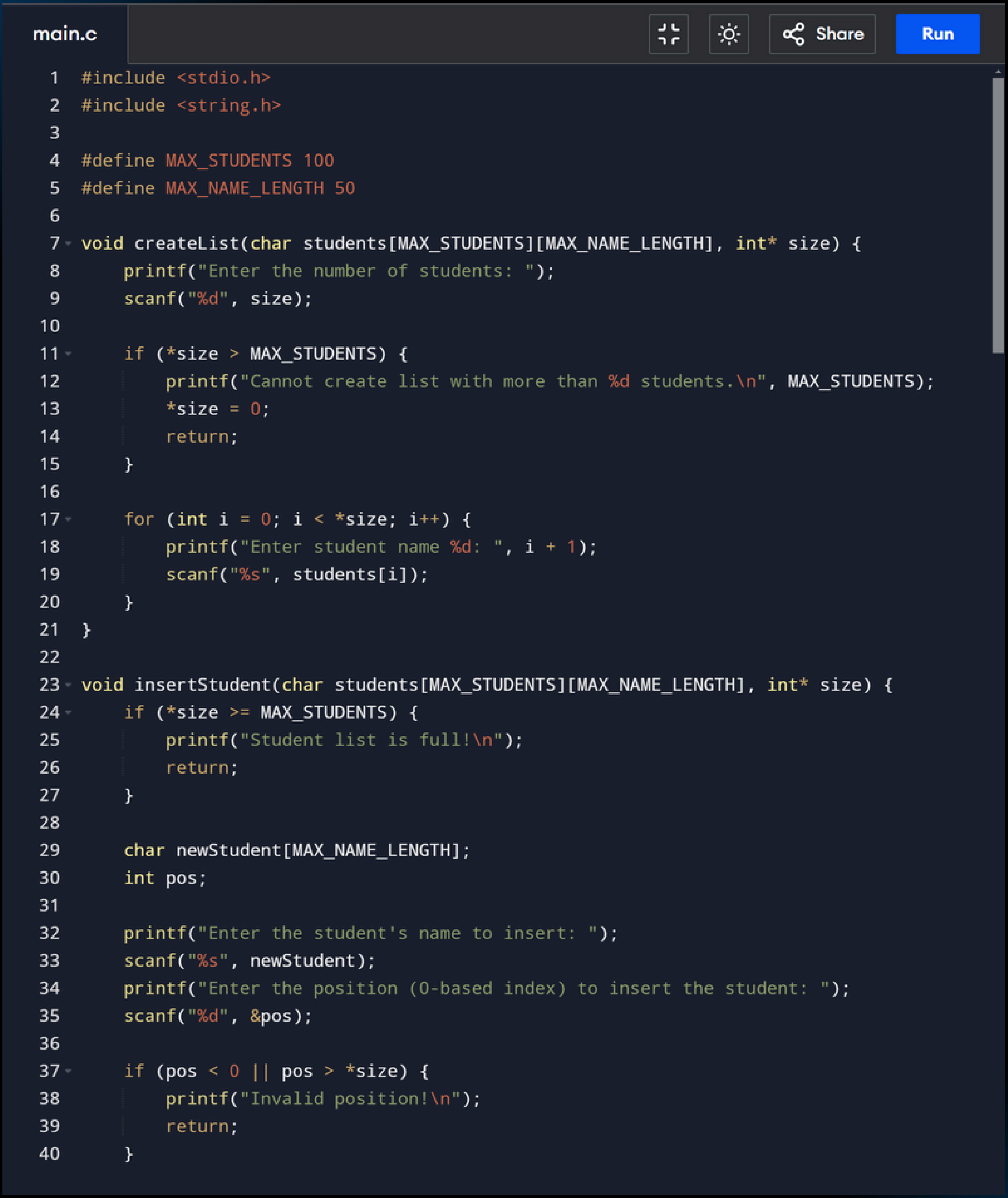
```c
    int** mat2 = allocateMatrix(rows2, cols2);

    // Input elements of the first matrix
    printf("Matrix 1:\n");
    inputMatrix(mat1, rows1, cols1);

    // Input elements of the second matrix
    printf("Matrix 2:\n");
    inputMatrix(mat2, rows2, cols2);

    // Multiply the matrices
    int** result = multiplyMatrices(mat1, rows1, cols1, mat2, rows2,
        cols2);

    // Print the result
    if (result != NULL) {
        printf("Resultant Matrix:\n");
        printMatrix(result, rows1, cols2);

        // Free the result matrix
        freeMatrix(result, rows1);
    }

    // Free dynamically allocated memory for the input matrices
    freeMatrix(mat1, rows1);
    freeMatrix(mat2, rows2);

```



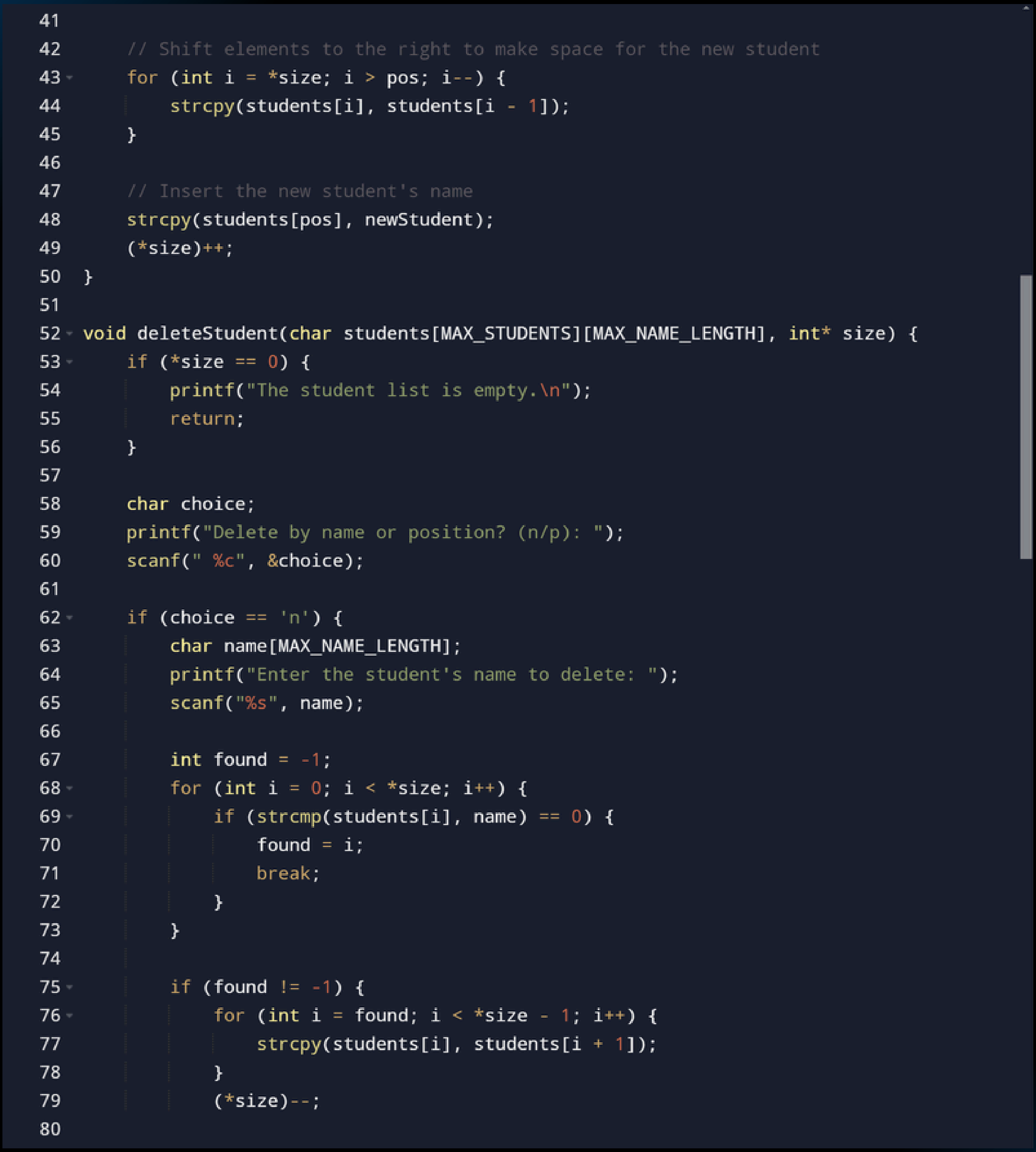
```
/tmp/2LWFP2tUgm.o
Enter the number of rows and columns for the first matrix: 3
3
Enter the number of rows and columns for the second matrix: 3
3
Matrix 1:
Enter the elements of the matrix:
23
62
59
26
25
26
82
26
23
Matrix 2:
Enter the elements of the matrix:
84
256
26
26
25
25
25
25
23
Resultant Matrix:
5019 8913 3505
3484 7931 1899
8139 22217 3311
```

2. You are given a task with creating a simple student management system using arrays that will allow the user to manage student names. Implement the following operations on a list of student names using switch-case and arrays. After every operation, display the current list of students.

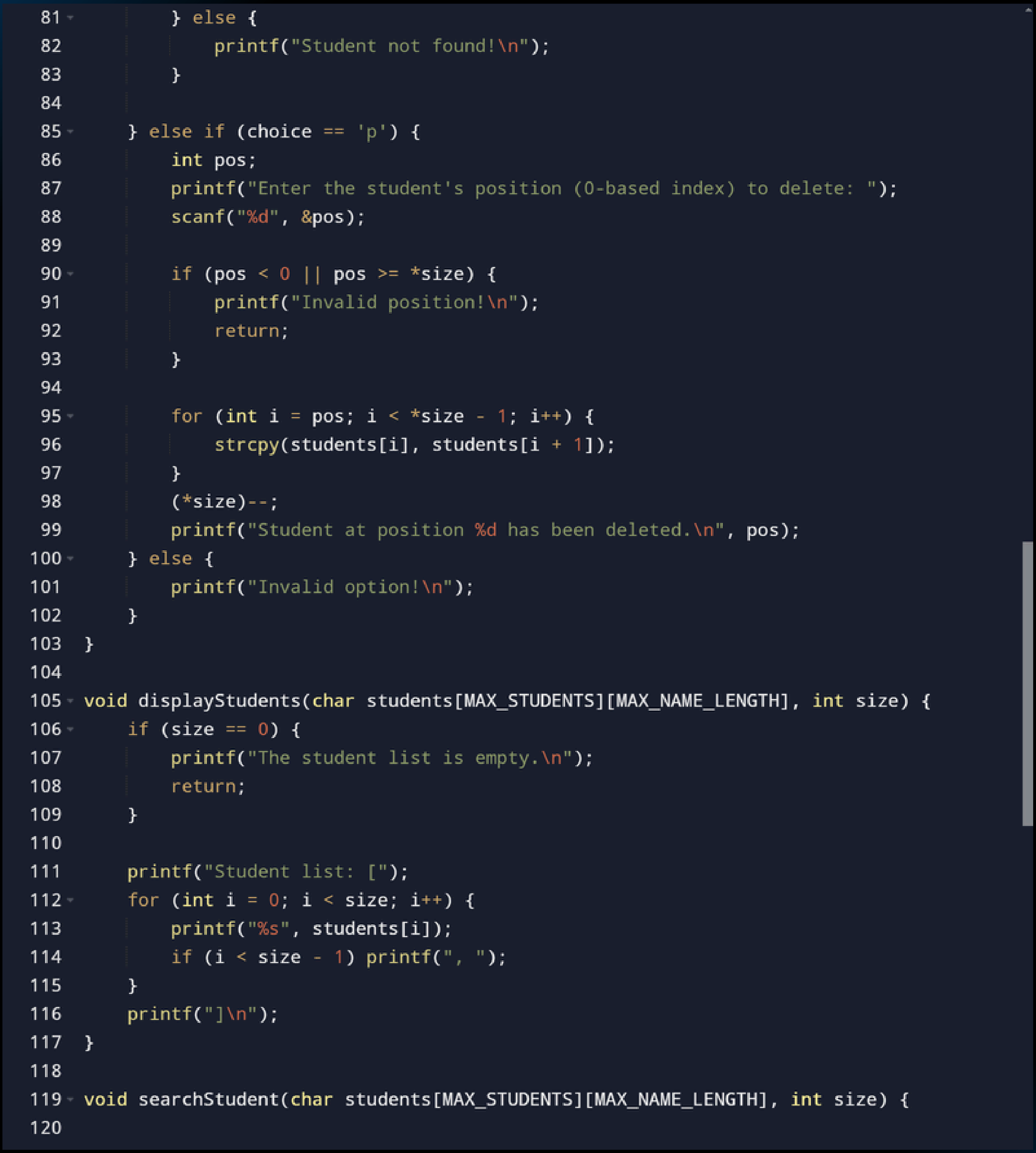
The operations to implement are:
(i) Creation of the list: Allow the user to create a list of student names by entering them one by one.
(ii) Insertion of a new student: Insert a new student's name into a specific position
in the list. The user should provide the name and the index at which it should be inserted.
(iii) Deletion of a student: Delete a student's name from the list based on their
position or name. Ask the user whether they want to delete by name or by index.
(iv) Traversal of the list: Display all the student names in the current order.
(v) Search for a student: Search for a student's name in the list and display
whether or not the student is found, along with their position if present.



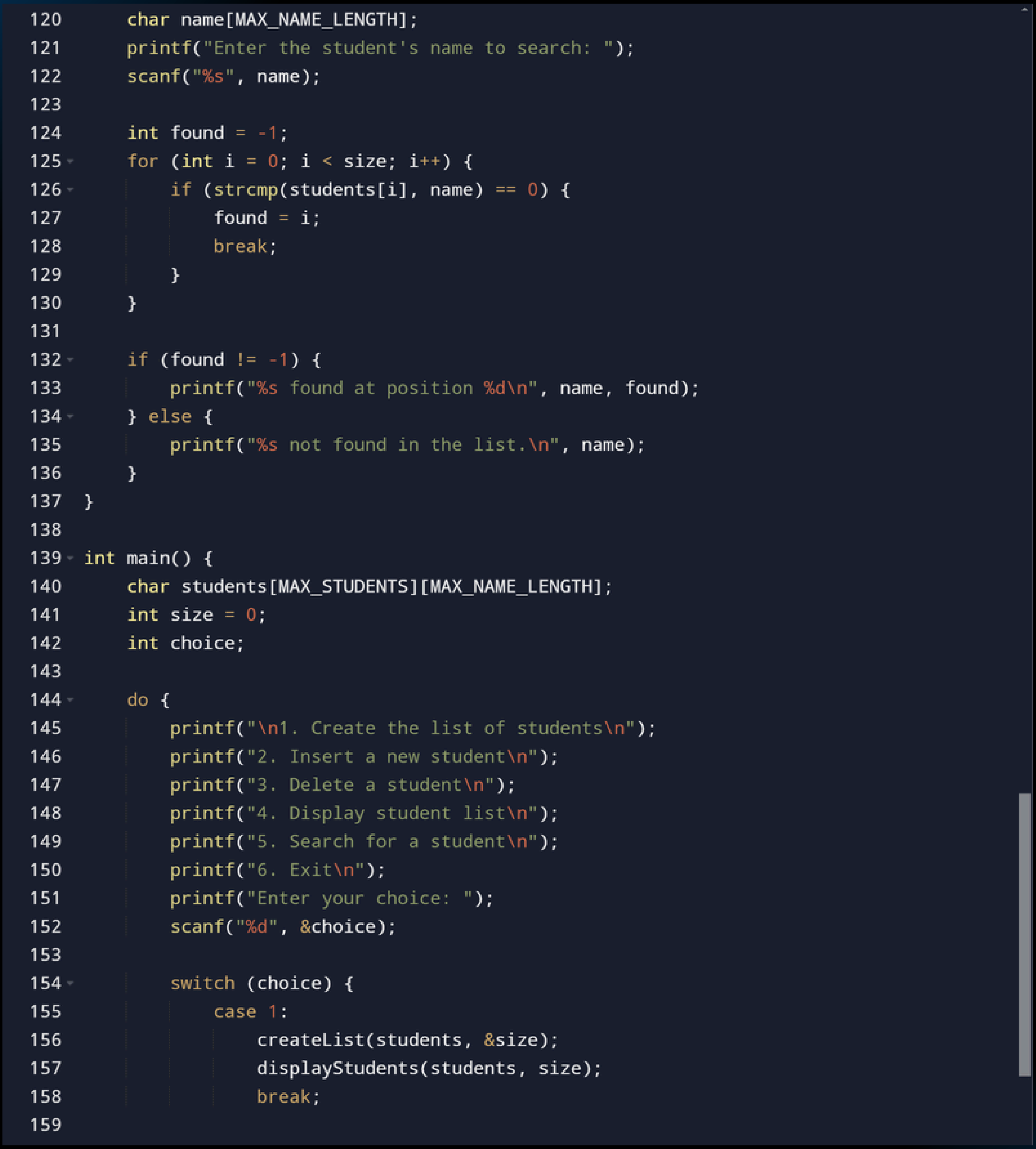
```c
#include <stdio.h>
#include <string.h>

#define MAX_STUDENTS 100
#define MAX_NAME_LENGTH 50

void createList(char students[MAX_STUDENTS][MAX_NAME_LENGTH], int* size) {
    printf("Enter the number of students: ");
    scanf("%d", size);

    if (*size > MAX_STUDENTS) {
        printf("Cannot create list with more than %d students.\n", MAX_STUDENTS);
        *size = 0;
        return;
    }

    for (int i = 0; i < *size; i++) {
        printf("Enter student name %d: ", i + 1);
        scanf("%s", students[i]);
    }
}

void insertStudent(char students[MAX_STUDENTS][MAX_NAME_LENGTH], int* size) {
    if (*size >= MAX_STUDENTS) {
        printf("Student list is full!\n");
        return;
    }

    char newStudent[MAX_NAME_LENGTH];
    int pos;

    printf("Enter the student's name to insert: ");
    scanf("%s", newStudent);
    printf("Enter the position (0-based index) to insert the student: ");
    scanf("%d", &pos);

    if (pos < 0 || pos > *size) {
        printf("Invalid position!\n");
        return;
    }
```



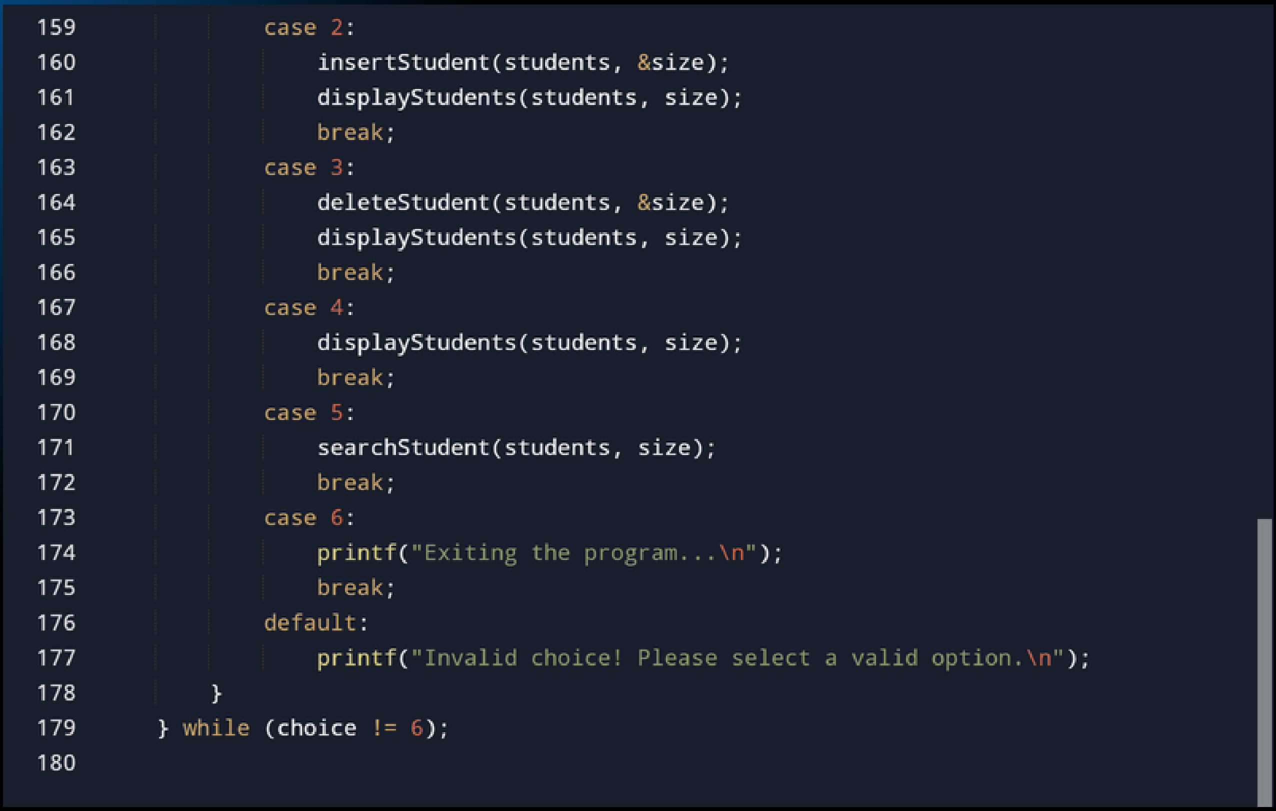
```c

    // Shift elements to the right to make space for the new student
    for (int i = *size; i > pos; i--) {
        strcpy(students[i], students[i - 1]);
    }

    // Insert the new student's name
    strcpy(students[pos], newStudent);
    (*size)++;
}

void deleteStudent(char students[MAX_STUDENTS][MAX_NAME_LENGTH], int* size) {
    if (*size == 0) {
        printf("The student list is empty.\n");
        return;
    }

    char choice;
    printf("Delete by name or position? (n/p): ");
    scanf(" %c", &choice);

    if (choice == 'n') {
        char name[MAX_NAME_LENGTH];
        printf("Enter the student's name to delete: ");
        scanf("%s", name);

        int found = -1;
        for (int i = 0; i < *size; i++) {
            if (strcmp(students[i], name) == 0) {
                found = i;
                break;
            }
        }

        if (found != -1) {
            for (int i = found; i < *size - 1; i++) {
                strcpy(students[i], students[i + 1]);
            }
            (*size)--;

```



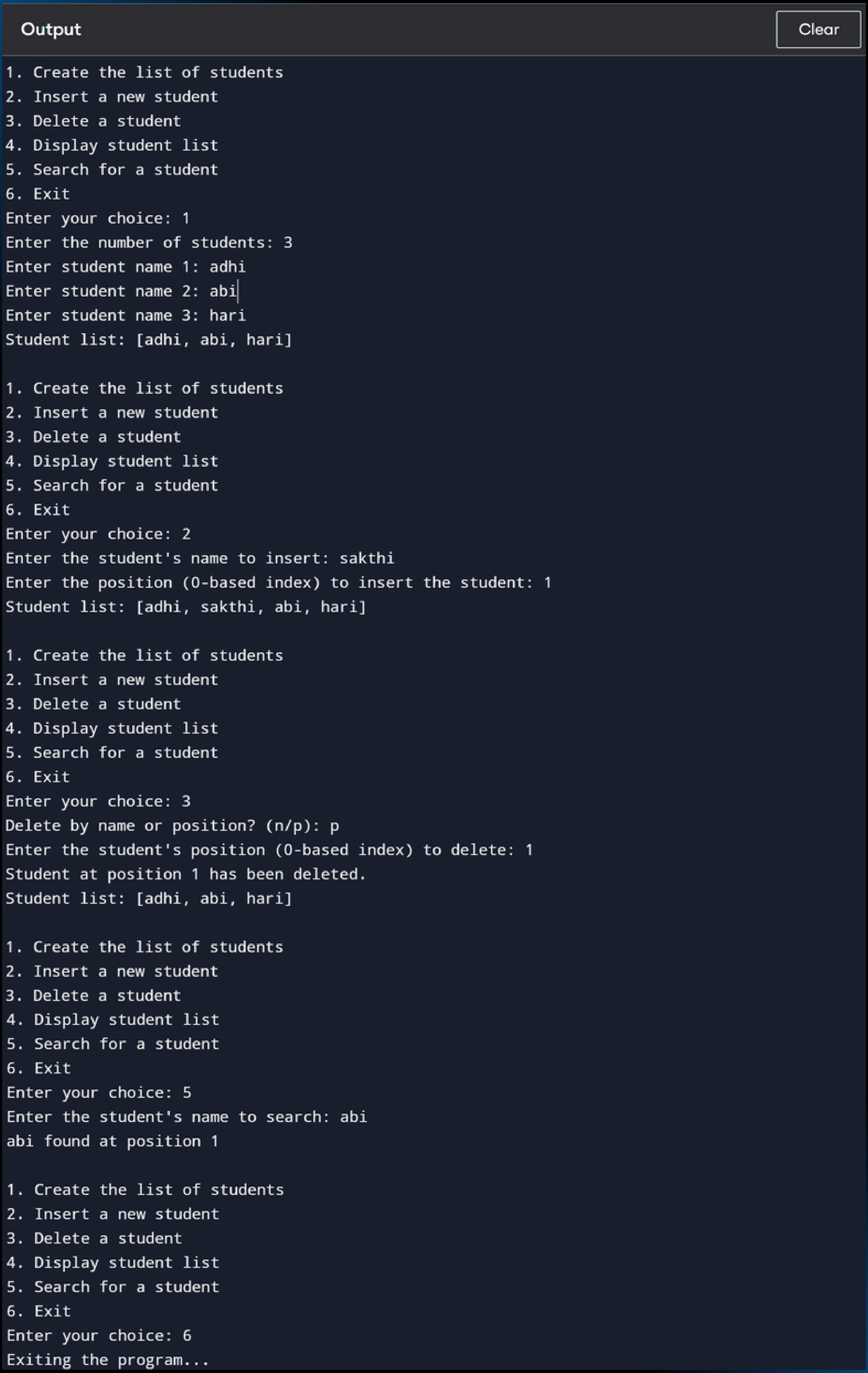
```c
        } else {
            printf("Student not found!\n");
        }

    } else if (choice == 'p') {
        int pos;
        printf("Enter the student's position (0-based index) to delete: ");
        scanf("%d", &pos);

        if (pos < 0 || pos >= *size) {
            printf("Invalid position!\n");
            return;
        }

        for (int i = pos; i < *size - 1; i++) {
            strcpy(students[i], students[i + 1]);
        }
        (*size)--;
        printf("Student at position %d has been deleted.\n", pos);
    } else {
        printf("Invalid option!\n");
    }
}

void displayStudents(char students[MAX_STUDENTS][MAX_NAME_LENGTH], int size) {
    if (size == 0) {
        printf("The student list is empty.\n");
        return;
    }

    printf("Student list: [");
    for (int i = 0; i < size; i++) {
        printf("%s", students[i]);
        if (i < size - 1) printf(", ");
    }
    printf("]\n");
}

void searchStudent(char students[MAX_STUDENTS][MAX_NAME_LENGTH], int size) {

```



```c
    char name[MAX_NAME_LENGTH];
    printf("Enter the student's name to search: ");
    scanf("%s", name);

    int found = -1;
    for (int i = 0; i < size; i++) {
        if (strcmp(students[i], name) == 0) {
            found = i;
            break;
        }
    }

    if (found != -1) {
        printf("%s found at position %d\n", name, found);
    } else {
        printf("%s not found in the list.\n", name);
    }
}

int main() {
    char students[MAX_STUDENTS][MAX_NAME_LENGTH];
    int size = 0;
    int choice;

    do {
        printf("\n1. Create the list of students\n");
        printf("2. Insert a new student\n");
        printf("3. Delete a student\n");
        printf("4. Display student list\n");
        printf("5. Search for a student\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                createList(students, &size);
                displayStudents(students, size);
                break;

```



```c
            case 2:
                insertStudent(students, &size);
                displayStudents(students, size);
                break;
            case 3:
                deleteStudent(students, &size);
                displayStudents(students, size);
                break;
            case 4:
                displayStudents(students, size);
                break;
            case 5:
                searchStudent(students, size);
                break;
            case 6:
                printf("Exiting the program...\n");
                break;
            default:
                printf("Invalid choice! Please select a valid option.\n");
        }
    } while (choice != 6);

```



```
1. Create the list of students
2. Insert a new student
3. Delete a student
4. Display student list
5. Search for a student
6. Exit
Enter your choice: 1
Enter the number of students: 3
Enter student name 1: adhi
Enter student name 2: abi
Enter student name 3: hari
Student list: [adhi, abi, hari]

1. Create the list of students
2. Insert a new student
3. Delete a student
4. Display student list
5. Search for a student
6. Exit
Enter your choice: 2
Enter the student's name to insert: sakthi
Enter the position (0-based index) to insert the student: 1
Student list: [adhi, sakthi, abi, hari]

1. Create the list of students
2. Insert a new student
3. Delete a student
4. Display student list
5. Search for a student
6. Exit
Enter your choice: 3
Delete by name or position? (n/p): p
Enter the student's position (0-based index) to delete: 1
Student at position 1 has been deleted.
Student list: [adhi, abi, hari]

1. Create the list of students
2. Insert a new student
3. Delete a student
4. Display student list
5. Search for a student
6. Exit
Enter your choice: 5
Enter the student's name to search: abi
abi found at position 1

1. Create the list of students
2. Insert a new student
3. Delete a student
4. Display student list
5. Search for a student
6. Exit
Enter your choice: 6
Exiting the program...
```