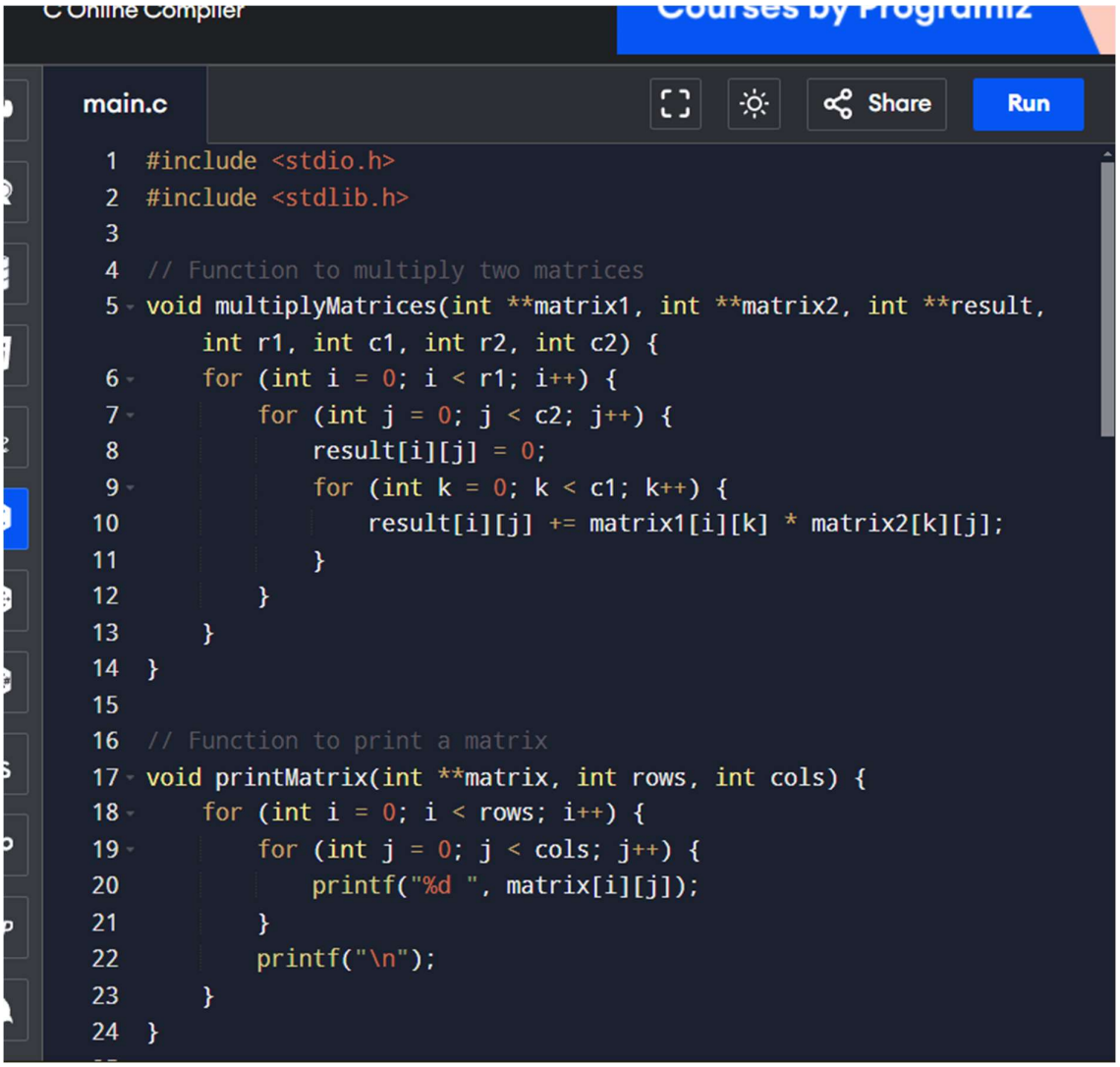
DSA ASSIGNMENT 2

Parvathy OK

RA2311026050020

QUESTION 1:-




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

// Function to multiply two matrices
void multiplyMatrices(int **matrix1, int **matrix2, int **result,
    int r1, int c1, int r2, int c2) {
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            result[i][j] = 0;
            for (int k = 0; k < c1; k++) {
                result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }
}

// 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]);
        }
        printf("\n");
    }
}
```

main.c

Share Run

```c

int main() {
    int r1, c1, r2, c2;

    // Get the dimensions of the first matrix
    printf("Enter the number of rows for the first matrix: ");
    scanf("%d", &r1);
    printf("Enter the number of columns for the first matrix: ");
    scanf("%d", &c1);

    // Get the dimensions of the second matrix
    printf("Enter the number of rows for the second matrix: ");
    scanf("%d", &r2);
    printf("Enter the number of columns for the second matrix: ");
    scanf("%d", &c2);

    // Check if the matrices can be multiplied
    if (c1 != r2) {
        printf("The matrices cannot be multiplied.\n");
        return 1;
    }

    // Dynamically allocate memory for the matrices
    int **matrix1 = (int **)malloc(r1 * sizeof(int *));
    for (int i = 0; i < r1; i++) {
        matrix1[i] = (int *)malloc(c1 * sizeof(int));
```

```c
    for (int i = 0; i < r1; i++) {
        matrix1[i] = (int *)malloc(c1 * sizeof(int));
    }

    int **matrix2 = (int **)malloc(r2 * sizeof(int *));
    for (int i = 0; i < r2; i++) {
        matrix2[i] = (int *)malloc(c2 * sizeof(int));
    }

    int **result = (int **)malloc(r1 * sizeof(int *));
    for (int i = 0; i < r1; i++) {
        result[i] = (int *)malloc(c2 * sizeof(int));
    }

    // Get the elements of the first matrix
    printf("Enter the elements of the first matrix:\n");
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c1; j++) {
            scanf("%d", &matrix1[i][j]);
        }
    }

    // Get the elements of the second matrix
    printf("Enter the elements of the second matrix:\n");
    for (int i = 0; i < r2; i++) {
        for (int j = 0; j < c2; j++) {
```

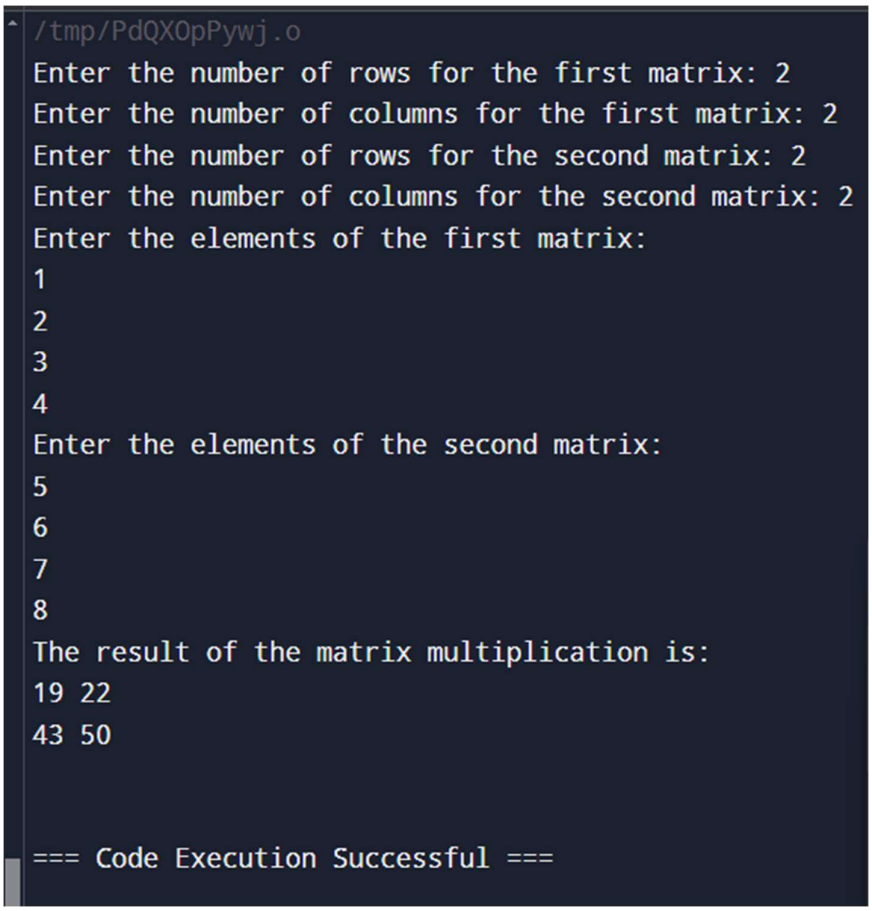
main.c

Share Run

```
74          for (int j = 0; j < c2; j++) {
75              scanf("%d", &matrix2[i][j]);
76          }
77      }
78
79      // Multiply the matrices
80      multiplyMatrices(matrix1, matrix2, result, r1, c1, r2, c2);
81
82      // Print the result
83      printf("The result of the matrix multiplication is:\n");
84      printMatrix(result, r1, c2);
85
86      // Free the dynamically allocated memory
87      for (int i = 0; i < r1; i++) {
88          free(matrix1[i]);
89      }
90      free(matrix1);
91
92      for (int i = 0; i < r2; i++) {
93          free(matrix2[i]);
94      }
95      free(matrix2);
96
97      for (int i = 0; i < r1; i++) {
98          free(result[i]);
99      }
```

```
98          free(result[i]);
99      }
100     free(result);
101
102     return 0;
103 }
```

SAMPLE OUTPUT:


/tmp/PdQXOpPywj.o
Enter the number of rows for the first matrix: 2
Enter the number of columns for the first matrix: 2
Enter the number of rows for the second matrix: 2
Enter the number of columns for the second matrix: 2
Enter the elements of the first matrix:
1
2
3
4
Enter the elements of the second matrix:
5
6
7
8
The result of the matrix multiplication is:
19 22
43 50
=== Code Execution Successful ===

QUESTION 2:-

main.c

Share Run

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

#define MAX_STUDENTS 100
#define MAX_NAME_LENGTH 50

char students[MAX_STUDENTS][MAX_NAME_LENGTH];
int num_students = 0;

void create_list() {
    printf("Enter the number of students: ");
    scanf("%d", &num_students);
    for (int i = 0; i < num_students; i++) {
        printf("Enter student %d name: ", i + 1);
        scanf("%s", students[i]);
    }
    printf("List created successfully!\n");
}

void insert_student() {
    char name[MAX_NAME_LENGTH];
    int index;
    printf("Enter the new student's name: ");
    scanf("%s", name);
    printf("Enter the index to insert the student: ");
```

```
    printf("Enter the index to insert the student: ");
    scanf("%d", &index);
    if (index < 0 || index > num_students) {
        printf("Invalid index. Please try again.\n");
    } else {
        for (int i = num_students; i > index; i--) {
            strcpy(students[i], students[i - 1]);
        }
        strcpy(students[index], name);
        num_students++;
        printf("Student inserted successfully!\n");
    }
}

void delete_student() {
    char choice;
    printf("Do you want to delete by name or by index? (name/index
        ): ");
    scanf(" %c", &choice);
    if (choice == 'n' || choice == 'N') {
        char name[MAX_NAME_LENGTH];
        printf("Enter the student's name to delete: ");
        scanf("%s", name);
        for (int i = 0; i < num_students; i++) {
            if (strcmp(students[i], name) == 0) {
                for (int j = i; j < num_students - 1; j++) {
                    strcpy(students[j], students[j + 1]);
```

```
                for (int j = i; j < num_students - 1; j++) {
                    strcpy(students[j], students[j + 1]);
                }
                num_students--;
                printf("Student deleted successfully!\n");
                return;
            }
        }
        printf("Student not found.\n");
    } else if (choice == 'i' || choice == 'I') {
        int index;
        printf("Enter the index to delete the student: ");
        scanf("%d", &index);
        if (index < 0 || index >= num_students) {
            printf("Invalid index. Please try again.\n");
        } else {
            for (int i = index; i < num_students - 1; i++) {
                strcpy(students[i], students[i + 1]);
            }
            num_students--;
            printf("Student deleted successfully!\n");
        }
    } else {
        printf("Invalid choice. Please try again.\n");
    }
}
```

```c
}

void traverse_list() {
    printf("Current list of students:\n");
    for (int i = 0; i < num_students; i++) {
        printf("%d. %s\n", i + 1, students[i]);
    }
}

void search_student() {
    char name[MAX_NAME_LENGTH];
    printf("Enter the student's name to search: ");
    scanf("%s", name);
    for (int i = 0; i < num_students; i++) {
        if (strcmp(students[i], name) == 0) {
            printf("Student found at position %d.\n", i + 1);
            return;
        }
    }
    printf("Student not found.\n");
}

int main() {
    while (1) {
        printf("\nStudent Management System Menu:\n");
        printf("1. Create list\n");
```

```
        printf("\nStudent Management System Menu:\n");
        printf("1. Create list\n");
        printf("2. Insert student\n");
        printf("3. Delete student\n");
        printf("4. Traverse list\n");
        printf("5. Search student\n");
        printf("6. Exit\n");
        int choice;
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                create_list();
                break;
            case 2:
                insert_student();
                break;
            case 3:
                delete_student();
                break;
            case 4:
                traverse_list();
                break;
            case 5:
                search_student();
                break;
            case 6:
                break;
            case 6:
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
        traverse_list();
    }
    return 0;
}
```

SAMPLE OUTPUT:-

```
/tmp/05Zd0UJQUc.o

Student Management System Menu:
1. Create list
2. Insert student
3. Delete student
4. Traverse list
5. Search student
6. Exit
Enter your choice: 1
Enter the number of students: 3
Enter student 1 name: john
Enter student 2 name: alice
Enter student 3 name: bob
List created successfully!
Current list of students:
1. john
2. alice
3. bob

Student Management System Menu:
1. Create list
2. Insert student
3. Delete student
4. Traverse list
5. Search student
```

```
4. Traverse list
5. Search student
6. Exit
Enter your choice: 2
Enter the new student's name: mike
Enter the index to insert the student: 1
Student inserted successfully!
Current list of students:
1. john
2. mike
3. alice
4. bob

Student Management System Menu:
1. Create list
2. Insert student
3. Delete student
4. Traverse list
5. Search student
6. Exit
Enter your choice: 3
Do you want to delete by name or by index? (name/index): name
Enter the student's name to delete: Student not found.
Current list of students:
1. john
2. mike
3. alice
```

```
2. mike
3. alice
4. bob

Student Management System Menu:
1. Create list
2. Insert student
3. Delete student
4. Traverse list
5. Search student
6. Exit
Enter your choice: 4
Current list of students:
1. john
2. mike
3. alice
4. bob
Current list of students:
1. john
2. mike
3. alice
4. bob

Student Management System Menu:
1. Create list
2. Insert student
```

```
1. Create list
2. Insert student
3. Delete student
4. Traverse list
5. Search student
6. Exit
Enter your choice: 5
Enter the student's name to search: alice
Student found at position 3.
Current list of students:
1. john
2. mike
3. alice
4. bob

Student Management System Menu:
1. Create list
2. Insert student
3. Delete student
4. Traverse list
5. Search student
6. Exit
Enter your choice: 6


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