DATA STRUCTURE AND ALGORITHMS

QUESTION-1

INPUT:

main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>
int* allocateMatrix(int rows, int cols) {
    return (int*)malloc(rows * cols * sizeof(int));
}
void inputMatrix(int* matrix, int rows, int cols) {
    for (int i = 0; i < rows * cols; i++) {
        printf("Enter element [%d]: ", i);
        scanf("%d", matrix + i);
    }
}
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 * cols + j));
        }
        printf("\n");
    }
}
int* multiplyMatrices(int* matrix1, int* matrix2, int rows1, int cols1, int
    cols2) {
    int* result = allocateMatrix(rows1, cols2);
    if (!result) {
        fprintf(stderr, "Memory allocation failed!\n");
        exit(EXIT_FAILURE);
    }
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols2; j++) {
            *(result + i * cols2 + j) = 0;
            for (int k = 0; k < cols1; k++) {
                *(result + i * cols2 + j) += *(matrix1 + i * cols1 + k) *
                    *(matrix2 + k * cols2 + j);
            }
        }
    }
    return result;
```

```c
}
int main() {
    int rows1, cols1, rows2, cols2;
    printf("Enter dimensions for the first matrix (rows cols): ");
    scanf("%d %d", &rows1, &cols1);
    printf("Enter dimensions for the second matrix (rows cols): ");
    scanf("%d %d", &rows2, &cols2);
    if (cols1 != rows2) {
        printf("Error: Incompatible dimensions for multiplication.\n");
        return 1;
    }
    int* matrix1 = allocateMatrix(rows1, cols1);
    int* matrix2 = allocateMatrix(rows2, cols2);
    if (!matrix1 || !matrix2) {
        fprintf(stderr, "Memory allocation failed!\n");
        return 1;
    }
    printf("Input for first matrix:\n");
    inputMatrix(matrix1, rows1, cols1);
    printf("Input for second matrix:\n");
    inputMatrix(matrix2, rows2, cols2);
    int* result = multiplyMatrices(matrix1, matrix2, rows1, cols1, cols2);
    printf("Result of multiplication:\n");
    printMatrix(result, rows1, cols2);
    free(matrix1);
    free(matrix2);
    free(result);
    return 0;
}

```

OUTPUT:

Output Clear

```
/tmp/6FCCacyxJC.o
Enter dimensions for the first matrix (rows cols): 2 2
Enter dimensions for the second matrix (rows cols): 2 2
Input for first matrix:
Enter element [0]: 2
Enter element [1]: 4
Enter element [2]: 1
Enter element [3]: 5
Input for second matrix:
Enter element [0]: 6
Enter element [1]: 2
Enter element [2]: 3
Enter element [3]: 2
Result of multiplication:
24 12
21 12


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

QUESTION-2

INPUT:

```c
#include <stdio.h>
#include <string.h>
#define MAX_STUDENTS 100
#define NAME_LENGTH 50
void displayStudents(char students[][NAME_LENGTH], int count) {
    printf("Current list of students:\n");
    for (int i = 0; i < count; i++) {
        printf("%d: %s\n", i + 1, students[i]);
    }
}
int main() {
    char students[MAX_STUDENTS][NAME_LENGTH];
    int count = 0;
    int choice;
    do {
        printf("\nStudent Management System\n");
        printf("1. Create list of students\n");
        printf("2. Insert a new student\n");
        printf("3. Delete a student\n");
        printf("4. Traverse the list\n");
        printf("5. Search for a student\n");
        printf("0. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();
        switch (choice) {
            case 1: {
                printf("Enter the number of students to add: ");
                int n;
                scanf("%d", &n);
                getchar();
                for (int i = 0; i < n && count < MAX_STUDENTS; i++) {
                    printf("Enter student name %d: ", i + 1);
                    fgets(students[count], NAME_LENGTH, stdin);
                    students[count][strcspn(students[count], "\n")] = '\0';
                    count++;

                displayStudents(students, count);
                break;
            }
            case 2: {
                if (count >= MAX_STUDENTS) {
                    printf("List is full, cannot insert new student.\n");
                    break;
                }
                char newStudent[NAME_LENGTH];
                int index;
                printf("Enter the name of the new student: ");
                fgets(newStudent, NAME_LENGTH, stdin);
                newStudent[strcspn(newStudent, "\n")] = '\0';
                printf("Enter the position (1 to %d): ", count + 1);
                scanf("%d", &index);
                getchar();
                if (index < 1 || index > count + 1) {
                    printf("Invalid position!\n");
                } else {
                    for (int i = count; i >= index; i--) {
                        strcpy(students[i], students[i - 1]);
                    }
                    strcpy(students[index - 1], newStudent);
                    count++;
                    displayStudents(students, count);
                }
                break;
            }
            case 3: {
                printf("Delete by (1) Name or (2) Index? ");
                int deleteChoice;
                scanf("%d", &deleteChoice);
                getchar();
                if (deleteChoice == 1) {
                    char nameToDelete[NAME_LENGTH];

```

```
            fgets(nameToDelete, NAME_LENGTH, stdin);
            nameToDelete[strcspn(nameToDelete, "\n")] = '\0';
            int found = 0;
            for (int i = 0; i < count; i++) {
                if (strcmp(students[i], nameToDelete) == 0) {
                    found = 1;
                    for (int j = i; j < count - 1; j++) {
                        strcpy(students[j], students[j + 1]);
                    }
                    count--;
                    break;
                }
            }
            if (found) {
                printf("Student '%s' deleted.\n", nameToDelete);
            } else {
                printf("Student '%s' not found.\n", nameToDelete);
            }
        } else if (deleteChoice == 2) {
            int index;
            printf("Enter the position to delete (1 to %d): ", count);
            scanf("%d", &index);
            getchar();
            if (index < 1 || index > count) {
                printf("Invalid position!\n");
            } else {
                for (int i = index - 1; i < count - 1; i++) {
                    strcpy(students[i], students[i + 1]);
                }
                count--;
                printf("Student at position %d deleted.\n", index);
            }
        } else {
            printf("Invalid choice!\n");
        }
        displayStudents(students, count);

```

```
            break;
        }
        case 5: {
            char nameToSearch[NAME_LENGTH];
            printf("Enter the name of the student to search: ");
            fgets(nameToSearch, NAME_LENGTH, stdin);
            nameToSearch[strcspn(nameToSearch, "\n")] = '\0';
            int found = 0;
            for (int i = 0; i < count; i++) {
                if (strcmp(students[i], nameToSearch) == 0) {
                    printf("Student '%s' found at position %d.\n", nameToSearch, i + 1
                        );
                    found = 1;
                    break;
                }
            }
            if (!found) {
                printf("Student '%s' not found.\n", nameToSearch);
            }
            break;
        }
        case 0: {
            printf("Exiting...\n");
            break;
        }
        default: {
            printf("Invalid choice! Please try again.\n");
            break;
        }
    }
} while (choice != 0);
return 0;
}

```

OUTPUT:

Output Clear

```
/tmp/IzoJHKQGjz.o
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: sharan
Enter student name 2: ritish
Enter student name 3: surya
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: 3
Enter the position (0-based index) to insert the student: 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: 3
Delete by name or position? (n/p): 2
Invalid option!
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: 4
Student list: [sharan, 3, ritish, surya]
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: sharan
sharan found at position 0
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...


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

BY:

NAME: SHARAN JAISWAL A

REGN NO: RA2311026050238