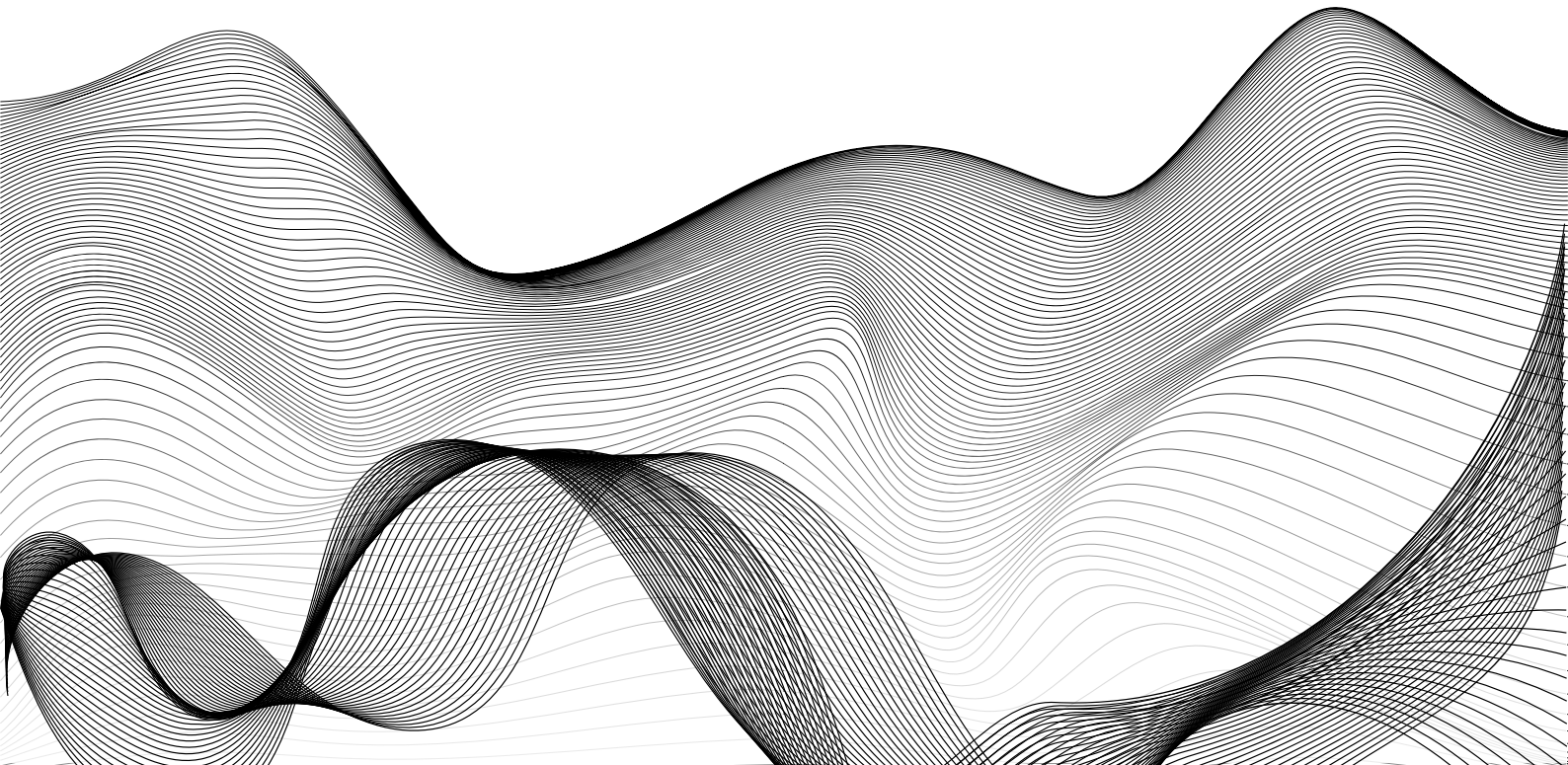
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)
TIRUCHIRAPPALLI

Prepared By: Zian Rajeshkumar Surani
RA2311026050085 - AIML B

Data Structure & Algorithm

Dynamic Memory Allocation and List

Date
22 September, 2024



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

ANSWER

```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 * cols * sizeof(int));
    return matrix;
}

// Function to deallocate memory for a matrix
void freemat(int* matrix) {
    free(matrix);
}

// Function to multiply two matrices
void multiplyMatrices(int* matrixA, int* matrixB, int* matrixC, int rowsA, int colsA, int colsB) {
    for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < colsB; j++) {
            matrixC[i * colsB + j] = 0;
            for (int k = 0; k < colsA; k++) {
                matrixC[i * colsB + j] += matrixA[i * colsA + k] * matrixB[k * colsB + 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 * cols + j]);
        }
        printf("\n");
    }
}
```

Note: Github link is in the github logo icon

```c
int main() {
    int rowsA, colsA, rowsB, colsB;

    // Get dimensions of matrices A and B
    printf("Enter number of rows for matrix A: ");
    scanf("%d", &rowsA);
    printf("Enter number of columns for matrix A: ");
    scanf("%d", &colsA);
    printf("Enter number of rows for matrix B: ");
    scanf("%d", &rowsB);
    printf("Enter number of columns for matrix B: ");
    scanf("%d", &colsB);

    // Check if matrices can be multiplied
    if (colsA != rowsB) {
        printf("Matrices cannot be multiplied.\n");
        return 1;
    }

    // Allocate memory for matrices A, B, and C
    int* matrixA = allocateMatrix(rowsA, colsA);
    int* matrixB = allocateMatrix(rowsB, colsB);
    int* matrixC = allocateMatrix(rowsA, colsB);

    // Initialize matrices A and B
    printf("Enter elements of matrix A:\n");
    for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < colsA; j++) {
            scanf("%d", &matrixA[i * colsA + j]);
        }
    }
    printf("Enter elements of matrix B:\n");
    for (int i = 0; i < rowsB; i++) {
        for (int j = 0; j < colsB; j++) {
            scanf("%d", &matrixB[i * colsB + j]);
        }
    }
    // Multiply matrices A and B
    multiplyMatrices(matrixA, matrixB, matrixC, rowsA, colsA, colsB);


    printf("Matrix A:\n");// Print result matrix
    printMatrix(matrixA, rowsA, colsA);
    printf("Matrix B:\n");
    printMatrix(matrixB, rowsB, colsB);
    printf("Matrix C (A * B):\n");
    printMatrix(matrixC, rowsA, colsB);

    // free matrix memory
    freemat(matrixA);
    freemat(matrixB);
    freemat(matrixC);

    return 0;
}
```

OUTPUT

```
Enter number of rows for matrix A: 3
Enter number of columns for matrix A: 3
Enter number of rows for matrix B: 3
Enter number of columns for matrix B: 3
Enter elements of matrix A:
1
2
3
4
5
6
7
8
9
Enter elements of matrix B:
9
8
7
6
5
4
3
2
Matrix A:
1 2 3
4 5 6
7 8 9
Matrix B:
9 8 7
6 5 4
3 2 1
Matrix C (A * B):
30 24 18
84 69 54
138 114 90
```

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.

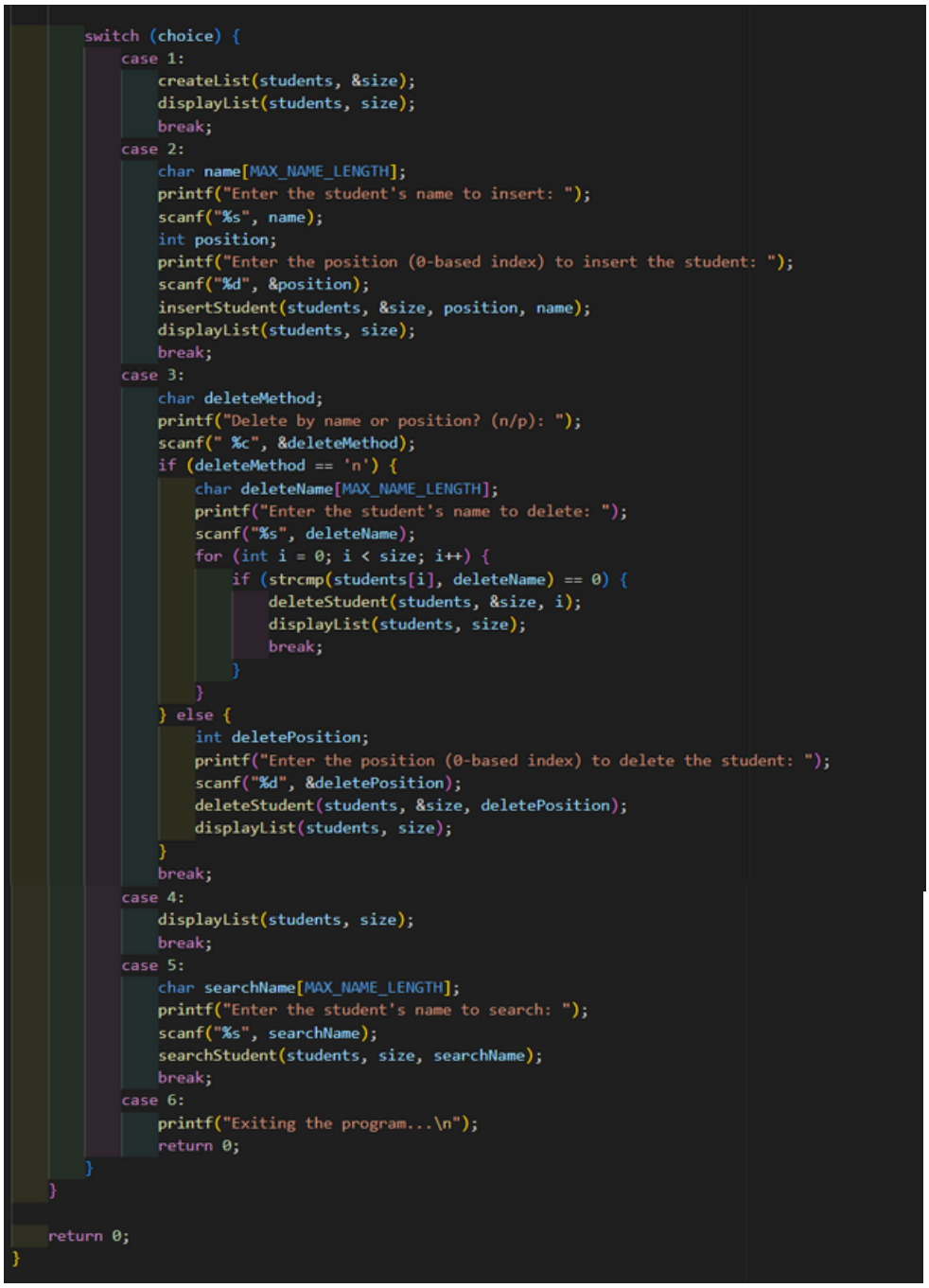
(iii) Deletion of a student: Delete a student's name from the list based on their position or name. Ask the user whether they want to delete by name or by index.

(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.

ANSWER

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

#define MAX_STUDENTS 100
#define MAX_NAME_LENGTH 50

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

void insertStudent(char students[][MAX_NAME_LENGTH], int* size, int position, char* name) {
    for (int i = *size; i > position; i--) {
        strcpy(students[i], students[i - 1]);
    }
    strcpy(students[position], name);
    (*size)++;
}

void deleteStudent(char students[][MAX_NAME_LENGTH], int* size, int position) {
    for (int i = position; i < *size - 1; i++) {
        strcpy(students[i], students[i + 1]);
    }
    (*size)--;
}

void displayList(char students[][MAX_NAME_LENGTH], int size) {
    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_NAME_LENGTH], int size, char* name) {
    int found = 0;
    for (int i = 0; i < size; i++) {
        if (strcmp(students[i], name) == 0) {
            printf("%s found at position %d\n", name, i);
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("%s not found\n", name);
    }
}

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

    while (1) {
        printf("1. 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: ");
        int choice;
        scanf("%d", &choice);
```



```
        switch (choice) {
            case 1:
                createList(students, &size);
                displayList(students, size);
                break;
            case 2:
                char name[MAX_NAME_LENGTH];
                printf("Enter the student's name to insert: ");
                scanf("%s", name);
                int position;
                printf("Enter the position (0-based index) to insert the student: ");
                scanf("%d", &position);
                insertStudent(students, &size, position, name);
                displayList(students, size);
                break;
            case 3:
                char deleteMethod;
                printf("Delete by name or position? (n/p): ");
                scanf(" %c", &deleteMethod);
                if (deleteMethod == 'n') {
                    char deleteName[MAX_NAME_LENGTH];
                    printf("Enter the student's name to delete: ");
                    scanf("%s", deleteName);
                    for (int i = 0; i < size; i++) {
                        if (strcmp(students[i], deleteName) == 0) {
                            deleteStudent(students, &size, i);
                            displayList(students, size);
                            break;
                        }
                    }
                } else {
                    int deletePosition;
                    printf("Enter the position (0-based index) to delete the student: ");
                    scanf("%d", &deletePosition);
                    deleteStudent(students, &size, deletePosition);
                    displayList(students, size);
                }
                break;
            case 4:
                displayList(students, size);
                break;
            case 5:
                char searchName[MAX_NAME_LENGTH];
                printf("Enter the student's name to search: ");
                scanf("%s", searchName);
                searchStudent(students, size, searchName);
                break;
            case 6:
                printf("Exiting the program...\n");
                return 0;
        }
    }

    return 0;
}
```

OUTPUT

```
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: 2
Enter student name 1: zian
Enter student name 2: ritu
Student list: [zian, ritu]
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: bhuppi
Enter the position (0-based index) to insert the student: 2
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): n
Enter the student's name to delete: ritu
Student list: [zian, bhuppi]
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: [zian, bhuppi]
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: zian
zian 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...
```