DSA ASSIGNMENT

Madhav K Mohan
RA2311026050003
AIML B

Dynamic Memory Allocation and List

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>

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; i++) {
        for (int j = 0; j < cols; j++) {
            printf("Enter element [%d][%d]: ", i, j);
            scanf("%d", (matrix + i * cols + j));
        }
    }
}
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);
    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;
int main() {
    int rows1, cols1, rows2, cols2;
    printf("Enter number of rows and columns for the first matrix: ");
    scanf("%d %d", &rows1, &cols1);
    printf("Enter number of rows and columns for the second matrix: ");
    scanf("%d %d", &rows2, &cols2);
    if (cols1 != rows2) {
        printf("Error: Matrix multiplication is not possible with the given dimensions.\n");
        return -1;
    }
    int* matrix1 = allocateMatrix(rows1, cols1);
    int* matrix2 = allocateMatrix(rows2, cols2);
    printf("Enter elements for the first matrix:\n");
    inputMatrix(matrix1, rows1, cols1);
    printf("Enter elements for the second matrix:\n");
    inputMatrix(matrix2, rows2, cols2);

    int* result = multiplyMatrices(matrix1, matrix2, rows1, cols1, cols2);

    printf("Resultant matrix after multiplication:\n");
    printMatrix(result, rows1, cols2);
    free(matrix1);
    free(matrix2);
    free(result);
    return 0;
}
```

Output:

```
Enter number of rows and columns for the first matrix: 3
3
Enter number of rows and columns for the second matrix: 3
3
Enter elements for the first matrix:
Enter element [0][0]: 2
Enter element [0][1]: 4
Enter element [0][2]: 6
Enter element [1][0]: 8
Enter element [1][1]: 10
Enter element [1][2]: 12
Enter element [2][0]: 14
Enter element [2][1]: 16
Enter element [2][2]: 18
Enter elements for the second matrix:
Enter element [0][0]: 1
Enter element [0][1]: 3
Enter element [0][2]: 5
Enter element [1][0]: 7
Enter element [1][1]: 9
Enter element [1][2]: 11
Enter element [2][0]: 13
Enter element [2][1]: 15
Enter element [2][2]: 17
Resultant matrix after multiplication:
108 132 156
234 294 354
360 456 552


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

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 100
#define NAME_LEN 50
void displayList(char students[MAX][NAME_LEN], int count) {
    if (count==0) {
        printf("The student list is empty.\n");
    } else {
        printf("Student list: [");
        for(int i=0;i<count;i++) {
            printf("%s", students[i]);
            if(i<count-1) {
                printf(",");
            }
        }
        printf("]\n");
    }
}
void createList(char students[MAX][NAME_LEN], int*count) {
    printf("Enter the number of students:");
    scanf("%d", count);
    getchar();
    for(int i=0;i<*count;i++) {
        printf("Enter student name %d:", i+1);
        fgets(students[i], NAME_LEN, stdin);
        students[i][strcspn(students[i], "\n")]=0;
    }
    displayList(students, *count);
}
void insertStudent(char students [MAX][NAME_LEN], int *count) {
    if (*count == MAX) {
        printf("Student list is full, cannot insert more students. \n");
        return;
    }
    char name[NAME_LEN];
    int pos;
    printf("Enter the student's name to insert: ");
    getchar();
    fgets(name, NAME_LEN, stdin);
    name [strcspn(name, "\n")] = 0;
    printf("Enter the position (0-based index) to insert the student: ");
    scanf ("%d", &pos);
    if (pos < 0 || pos > *count) {
```

```
44          printf("Invalid position. \n");
45          return;
46      }
47      for (int i = *count; i > pos; i--) {
48      strcpy(students[i], students[i - 1]);
49      }
50      strcpy(students[pos], name);
51      (*count)++;
52      displayList(students, *count);
53  }
54  void deleteStudent(char students [MAX][NAME_LEN], int *count) {
55      if (*count == 0) {
56          printf("The student list is empty. \n");
57          return;
58      }
59      char choice;
60      printf("Delete by name or position? (n/p): ");
61      getchar();
62      scanf("%c", &choice);
63      if (choice =='n') {
64          char name [NAME_LEN];
65          int found = 0;
66          printf("Enter the student's name to delete: ");
67          getchar();
68          fgets(name, NAME_LEN, stdin);
69          name [strcspn(name, "\n" )] = 0;
70          for (int i = 0;i < *count;i++) {
71              if (strcmp(students[i], name) == 0) {
72                  for (int j = i;j< *count - 1;j++) {
73                      strcpy(students[j], students[j + 1]);
74                  }
75                  (* count) --;
76                  found = 1;
77                  break;
78              }
79          }
80          if(!found) {
81              printf ("Student not found. \n");
82          }
83      }else if (choice == 'p') {
84          int pos;
85          printf("Enter the position to delete the student: ");
86          scanf ("%d", &pos);
```

```
87          if (pos < 0 || pos >= *count) {
88              printf("Invalid position.\n");
89              return;
90          }
91          for (int i = pos; i < *count - 1; i++) {
92              strcpy(students[i], students[i + 1]);
93          }
94          (*count) --;
95      } else {
96          printf("Invalid choice.\n");
97      }
98      displayList(students, *count);
99  }
100 void searchStudent (char students [MAX][NAME_LEN], int count) {
101     char name[NAME_LEN];
102     int found = 0;
103     if (count == 0) {
104         printf("The student list is empty.\n");
105         return;
106     }
107     printf("Enter the student's name to search: ");
108     getchar();
109     fgets(name, NAME_LEN, stdin);
110     name [strcspn(name, "\n" )] = 0;
111     for (int i = 0; i < count; i++) {
112         if (strcmp(students[i], name) == 0) {
113             printf("%s found at position %d\n", name, i);
114             found = 1;
115             break;
116         }
117     }
118     if (!found) {
119         printf("%s not found in the list.\n", name);
120     }
121 }
122 int main() {
123     char students[MAX][NAME_LEN];
124     int count = 0;
125     int choice;
126     do {
127         printf("\n1. Create the list of students\n");
128         printf("2. Insert a new student\n");
```

```c
        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, &count);
                break;
            case 2:
                insertStudent (students, &count);
                break;
            case 3:
                deleteStudent (students, &count);
                break;
            case 4:
                displayList(students, count);
                break;
            case 5:
                searchStudent(students, count);
                break;
            case 6:
                printf("Exiting the program...\n");
                break;
            default:
                printf("Invalid choice. Please try again. \n");
        }
    } while (choice != 6);
    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:3
Enter student name 1:Madhav
Enter student name 2:Aswin
Enter student name 3:Sujin
Student list: [Madhav,Aswin,Sujin]

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: Naveen
Enter the position (0-based index) to insert the student: 1
Student list: [Madhav,Naveen,Aswin,Sujin]

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 position to delete the student: 1
Student list: [Madhav,Aswin,Sujin]
```

```
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: [Madhav,Aswin,Sujin]

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: Sujin
Sujin found at position 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: 6
Exiting the program...


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