DSA ASSIGNMENT

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

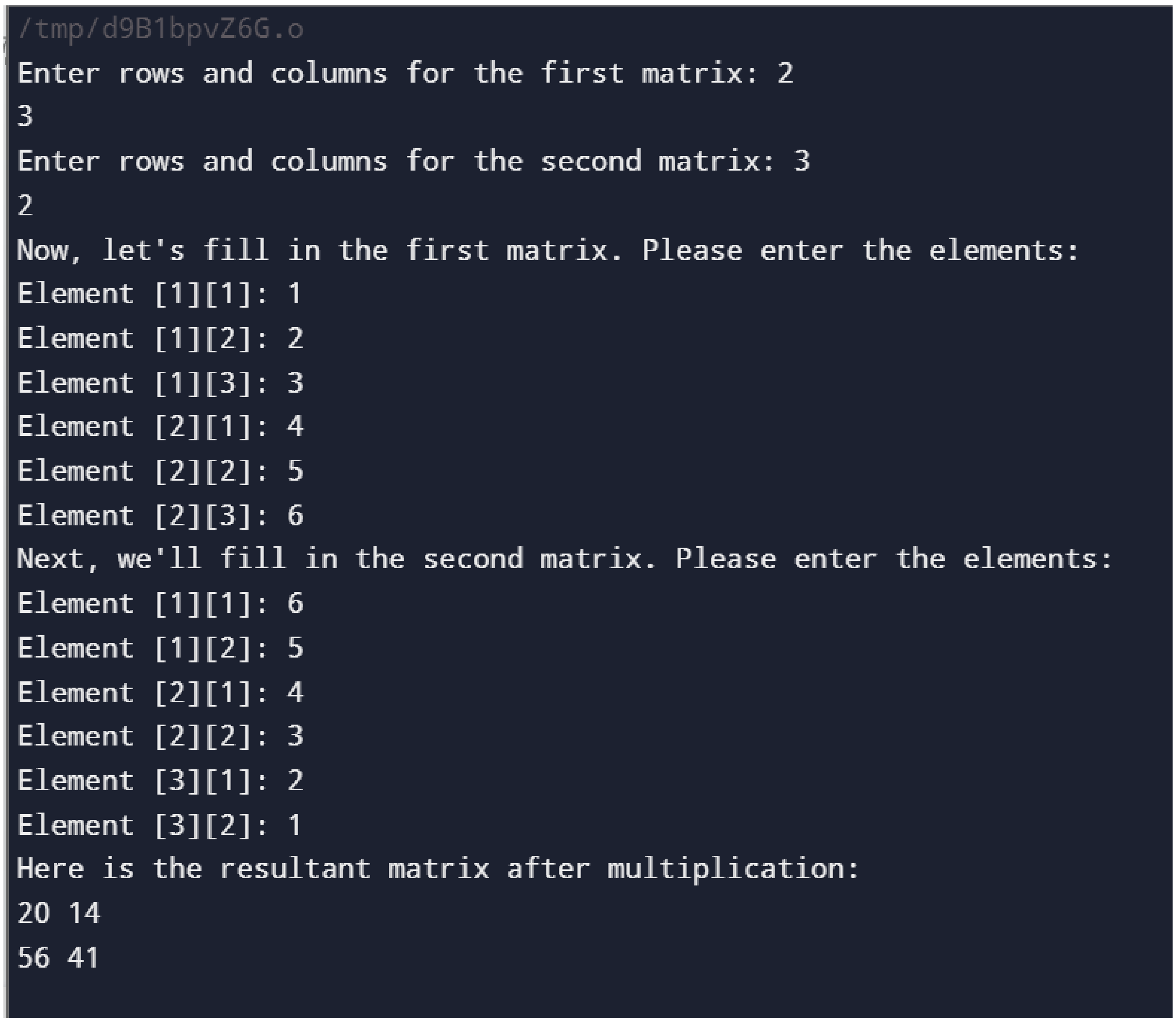
```
#include <stdio.h>
#include <stdlib.h>

int main() {
  int i, j, k;
  int row1, col1, row2, col2;

  // asking for the number of rows and columns of the first matrix
  printf("Enter rows and columns for the first matrix: ");
  scanf("%d %d", &row1, &col1);

  // asking for the number of rows and columns of the second matrix
  printf("Enter rows and columns for the second matrix: ");
  scanf("%d %d", &row2, &col2);

  // checking if the matrices can be multiplied
  if (col1 != row2) {
    printf("Matrix multiplication not possible because the number of columns in the first matrix "
        "must equal the number of rows in the second matrix.\n");
    return 1;
  }

  // allocating memory dynamically for the first matrix
  int **matrix1 = (int **)malloc(row1 * sizeof(int *));
  for (i = 0; i < row1; i++)
```

```
    matrix1[i] = (int *)malloc(col1 * sizeof(int));

// Allocating memory dynamically for the second matrix
int **matrix2 = (int **)malloc(row2 * sizeof(int *));
for (i = 0; i < row2; i++)
    matrix2[i] = (int *)malloc(col2 * sizeof(int));

// allocating memory dynamically for the result matrix
int **result = (int **)malloc(row1 * sizeof(int *));
for (i = 0; i < row1; i++)
    result[i] = (int *)malloc(col2 * sizeof(int));

// taking input elements for the first matrix
printf("Now, let's fill in the first matrix. Please enter the elements:\n");
for (i = 0; i < row1; i++) {
    for (j = 0; j < col1; j++) {
        printf("Element [%d][%d]: ", i + 1, j + 1); // prompting with element position
        scanf("%d", (*(matrix1 + i) + j));
    }
}

// taking input elements for the second matrix
printf("Next, we'll fill in the second matrix. Please enter the elements:\n");
for (i = 0; i < row2; i++) {
    for (j = 0; j < col2; j++) {
        printf("Element [%d][%d]: ", i + 1, j + 1); // prompting with element position
        scanf("%d", (*(matrix2 + i) + j));
    }
}

// initializing the result matrix to zero
```

```
for (i = 0; i < row1; i++)
    for (j = 0; j < col2; j++)
        *(*(result + i) + j) = 0;

// matrix multiplication
for (i = 0; i < row1; i++) {
    for (j = 0; j < col2; j++) {
        for (k = 0; k < col1; k++) {
            *(*(result + i) + j) += *(*(matrix1 + i) + k) * *(*(matrix2 + k) + j);
        }
    }
}

// displaying the resulting matrix
printf("Here is the resultant matrix after multiplication:\n");
for (i = 0; i < row1; i++) {
    for (j = 0; j < col2; j++) {
        printf("%d ", *(*(result + i) + j));
    }
    printf("\n");
}

// free allocated memory
for (i = 0; i < row1; i++)
    free(matrix1[i]);
free(matrix1);

for (i = 0; i < row2; i++)
    free(matrix2[i]);
free(matrix2);
```

```
    for (i = 0; i < row1; i++)
        free(result[i]);
    free(result);

    return 0;
}
```

OUTPUT:



```
/tmp/d9B1bpvZ6G.o
Enter rows and columns for the first matrix: 2
3
Enter rows and columns for the second matrix: 3
2
Now, let's fill in the first matrix. Please enter the elements:
Element [1][1]: 1
Element [1][2]: 2
Element [1][3]: 3
Element [2][1]: 4
Element [2][2]: 5
Element [2][3]: 6
Next, we'll fill in the second matrix. Please enter the elements:
Element [1][1]: 6
Element [1][2]: 5
Element [2][1]: 4
Element [2][2]: 3
Element [3][1]: 2
Element [3][2]: 1
Here is the resultant matrix after multiplication:
20 14
56 41
```

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.

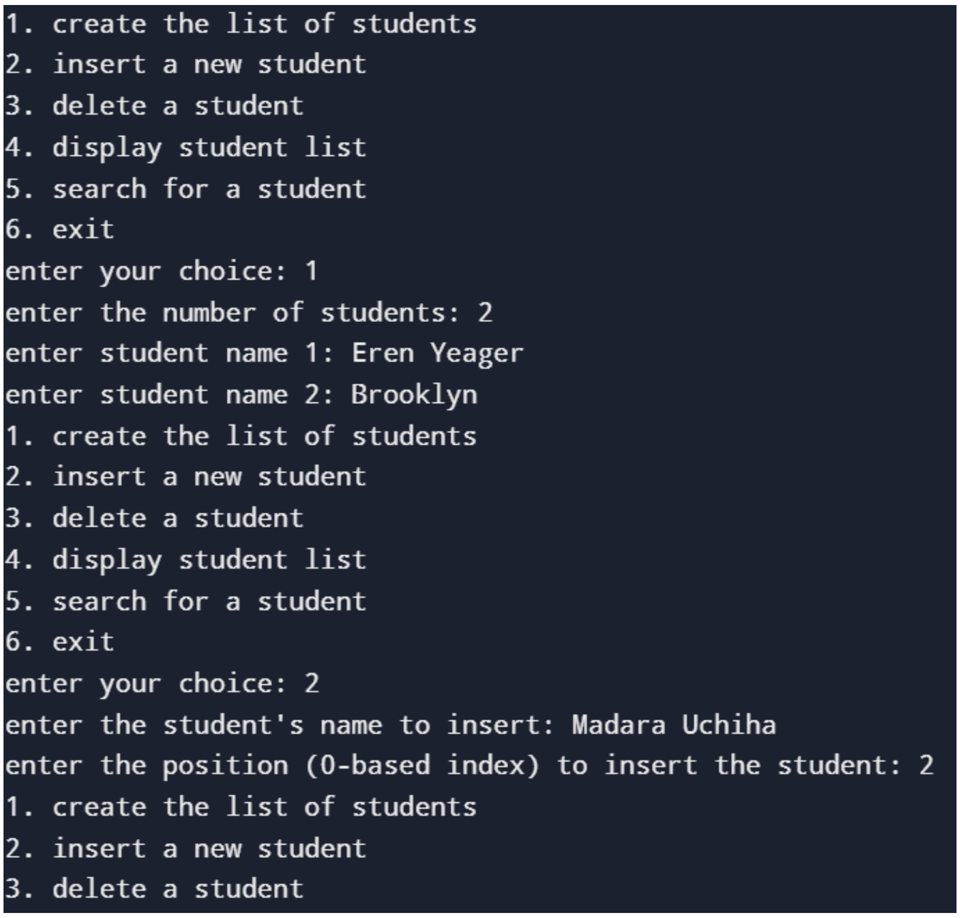
CODE:

```
#include <stdio.h>

#include <string.h>


int main() {

   const int max_students = 100; // maximum number of students

   const int name_length = 50; // maximum length of a student name


   char students[max_students][name_length]; // array to hold student names

   int count = 0; // current number of students

   int choice = 0;


   while (choice != 6) {

      // display the menu

      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: ");
scanf("%d", &choice);
getchar(); // consume newline character after entering choice

switch (choice) {
   case 1: {
      // creation of the list
      printf("enter the number of students: ");
      scanf("%d", &count);
      getchar(); // consume newline character
      for (int i = 0; i < count; i++) {
         printf("enter student name %d: ", i + 1);
         fgets(students[i], name_length, stdin);
         students[i][strcspn(students[i], "\n")] = 0; // remove newline character
      }
      break;
   }
   case 2: {
      // insertion of a new student
      if (count < max_students) {
         char new_student[name_length];
         int position;

         printf("enter the student's name to insert: ");
         fgets(new_student, name_length, stdin);
         new_student[strcspn(new_student, "\n")] = 0; // remove newline
```

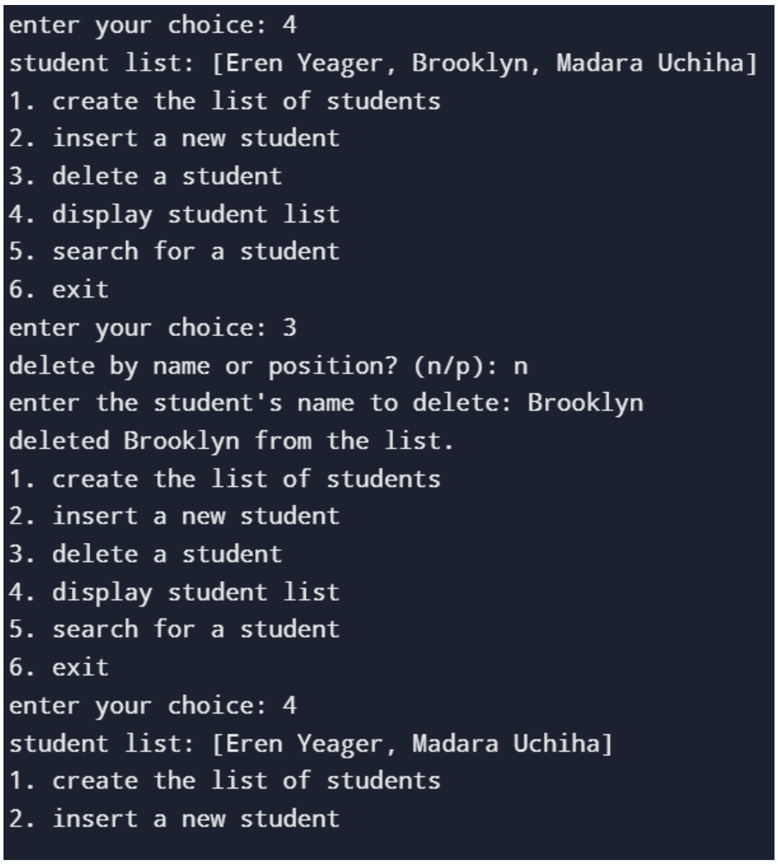
```
        printf("enter the position (0-based index) to insert the student: ");
        scanf("%d", &position);
        getchar(); // consume newline

        if (position >= 0 && position <= count) {
            for (int i = count; i > position; i--) {
                strcpy(students[i], students[i - 1]);
            }
            strcpy(students[position], new_student);
            count++;
        } else {
            printf("invalid position!\n");
        }
    } else {
        printf("cannot insert more students, list is full.\n");
    }
    break;
}
case 3: {
    // deletion of a student
    char delete_option;
    printf("delete by name or position? (n/p): ");
    scanf(" %c", &delete_option);
    getchar(); // consume newline

    if (delete_option == 'n') {
        char name_to_delete[name_length];
        printf("enter the student's name to delete: ");
        fgets(name_to_delete, name_length, stdin);
        name_to_delete[strcspn(name_to_delete, "\n")] = 0; // remove newline
        int found = 0;
        for (int i = 0; i < count; i++) {
            if (strcmp(students[i], name_to_delete) == 0) {
                found = 1;
                for (int j = i; j < count - 1; j++) {
                    strcpy(students[j], students[j + 1]);
                }
                count--;
                break;
            }
        }
        if (found) {
            printf("deleted %s from the list.\n", name_to_delete);
        } else {
            printf("student not found.\n");
        }
    } else if (delete_option == 'p') {
        int position;
        printf("enter the position (0-based index) to delete the student: ");
        scanf("%d", &position);
        getchar(); // consume newline

        if (position >= 0 && position < count) {
            for (int i = position; i < count - 1; i++) {
                strcpy(students[i], students[i + 1]);
            }
            count--;
            printf("deleted student at position %d.\n", position);
        } else {
            printf("invalid position!\n");
        }
    } else {
        printf("invalid option!\n");
    }
    break;
}
case 4: {
    // display student list
    printf("student list: [");
    for (int i = 0; i < count; i++) {
        printf("%s", students[i]);
        if (i < count - 1) {
            printf(", ");
        }
    }
    printf("]\n");
    break;
}
case 5: {
    // search for a student
    char name_to_search[name_length];
    printf("enter the student's name to search: ");
    fgets(name_to_search, name_length, stdin);
    name_to_search[strcspn(name_to_search, "\n")] = 0; // remove newline

    int found = 0;
    for (int i = 0; i < count; i++) {
        if (strcmp(students[i], name_to_search) == 0) {
            printf("%s found at position %d\n", name_to_search, i);
            found = 1;
            break;
        }
```

```
            }
            if (!found) {
                printf("%s not found.\n", name_to_search);
            }
            break;
        }
        case 6:
            printf("exiting the program...\n");
            break;
        default:
            printf("invalid choice! please enter a valid option.\n");
    }
  }

  return 0;
}
```

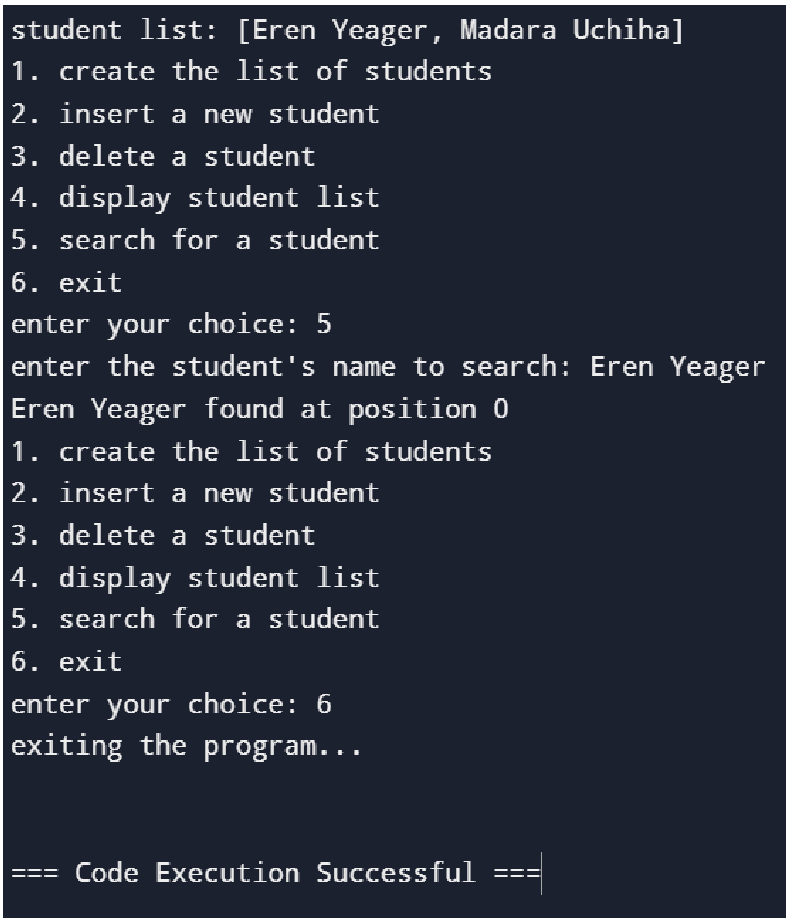
OUTPUT:



```
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: Eren Yeager
enter student name 2: Brooklyn
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: Madara Uchiha
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
```



```
enter your choice: 4
student list: [Eren Yeager, Brooklyn, Madara Uchiha]
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: Brooklyn
deleted Brooklyn from the list.
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: [Eren Yeager, Madara Uchiha]
1. create the list of students
2. insert a new student
```


```
student list: [Eren Yeager, Madara Uchiha]
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: Eren Yeager
Eren Yeager 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,

ANISH IDHAYAN I,

II – AIML-B,

RA2311026050078.