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

SRM Institute of Science & Technology

Department of Data Science & Business System

Title: STUDENT ENROLLMENT SYSTEM

21CSC201J-Data Structures and algorithm

- SAI TULASI ASRITHA SRI(RA2311027010110)

Problem statement

In an academic institution, efficiently managing student enrollments is crucial for maintaining organized records and supporting real-time access to student information.

The system should allow seamless additions of new students, quick search capabilities by student ID, and ensure records are sorted for easy report generation.

Core Requirement: The system must:

- Allow insertion of new students in real-time as they enroll.
- Enable easy and quick access to student records, particularly by student ID.
- Display the students enrolled in a specific course and the courses a specific student is enrolled in.

Data Selection-

Chosen Data Structure: Tree (Binary Search Tree).

- **Reason:**
    - A BST allows efficient organization of student records based on a key (e.g., student ID).
    - With the property of binary search, where each node has a left child with a smaller value and a right child with a larger value.
    - The BST structure enables fast lookups, insertions, and deletions in an organized manner.

JUSTIFICATION

- **Time Complexity**:
    - **Add Student (Insert Operation):** O(logn).
    - **Find Student (Search Operation):** O(logn).
    - **Remove Student (Delete Operation):** O(logn).
- **Space Complexity**:

    - O(n), where n is the total number of students. The BST uses memory space linearly with the number of enrolled students.

- **Justification**:
    - The Binary Search Tree structure supports quick retrieval of student records by ID.
    - Allowing the system to maintain real-time access to individual records for administrative and operational needs.

CODE TO SOLVE THE GIVEN PROBLEM

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


// Define a structure for a student

struct Student {

   int studentID;         // Unique ID for each student

   char name[100];        // Name of the student

   struct Student* next;    // Pointer to the next student in the list

};


// Function to create a new student node

struct Student* createStudent(int studentID, const char* name) {

   struct Student* newStudent = (struct Student*)malloc(sizeof(struct Student));

   newStudent->studentID = studentID;

   strcpy(newStudent->name, name); // Copy name to the student structure

   newStudent->next = NULL;

   return newStudent;

}
```

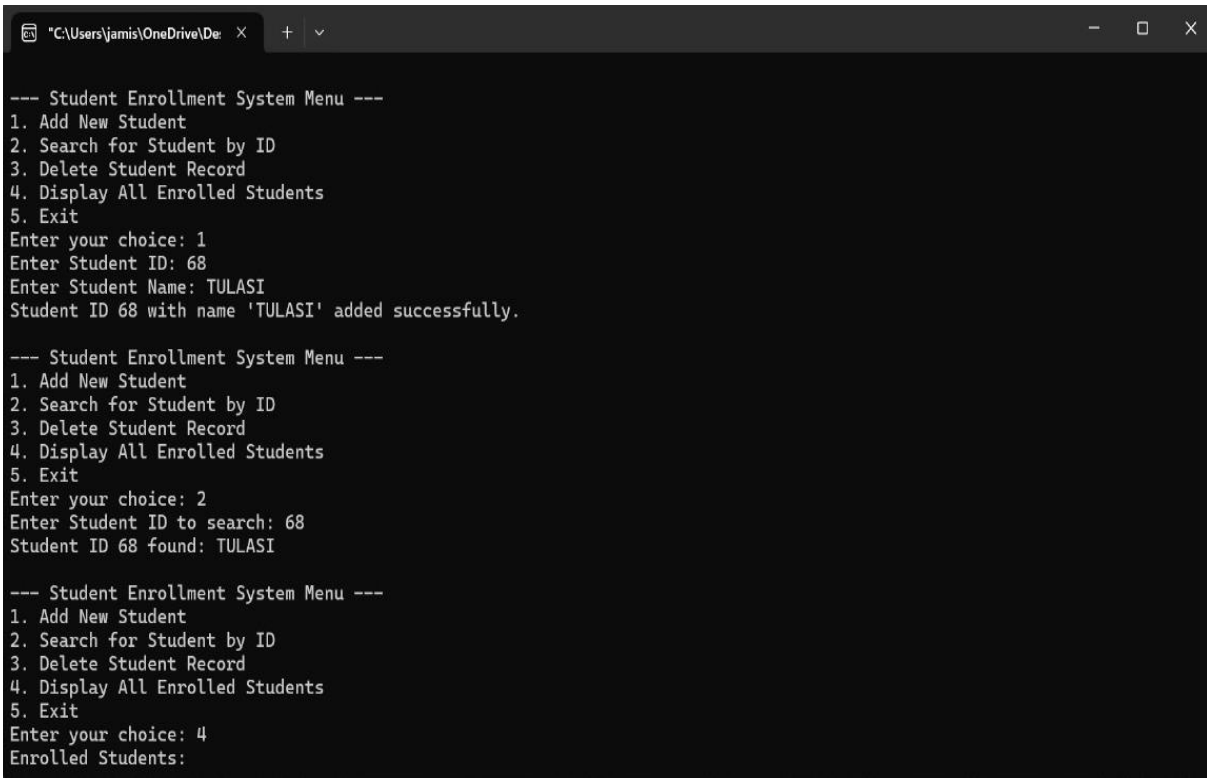
```
// Function to add a new student to the enrollment system
void addStudent(struct Student** head, int studentID, const char* name) {
  struct Student* newStudent = createStudent(studentID, name);
  newStudent->next = *head; // Add new student at the beginning of the list
  *head = newStudent;
  printf("Student ID %d with name '%s' added successfully.\n", studentID, name);
}

// Function to search for a student by ID
struct Student* searchStudent(struct Student* head, int studentID) {
  struct Student* current = head;
  while (current != NULL) {
    if (current->studentID == studentID) {
      return current; // Student found
    }
    current = current->next; // Move to the next student
  }
  return NULL; // Student not found
}

// Function to delete a student by ID
void deleteStudent(struct Student** head, int studentID) {
  struct Student* current = *head;
  struct Student* previous = NULL;

  while (current != NULL && current->studentID != studentID) {
    previous = current;
    current = current->next; // Move to the next student
  }
```

```
    if (current == NULL) {
        printf("Student ID %d not found.\n", studentID);
        return; // Student not found
    }

    if (previous == NULL) {
        // Deleting the first student in the list
        *head = current->next;
    } else {
        previous->next = current->next; // Bypass the current student
    }

    free(current); // Free the memory
    printf("Student ID %d deleted successfully.\n", studentID);
}

// Function to display all enrolled students
void displayStudents(struct Student* head) {
    if (head == NULL) {
        printf("No students enrolled.\n");
        return;
    }

    struct Student* current = head;
    printf("Enrolled Students:\n");
    while (current != NULL) {
        printf("Student ID: %d, Name: %s\n", current->studentID, current->name);
        current = current->next; // Move to the next student
```

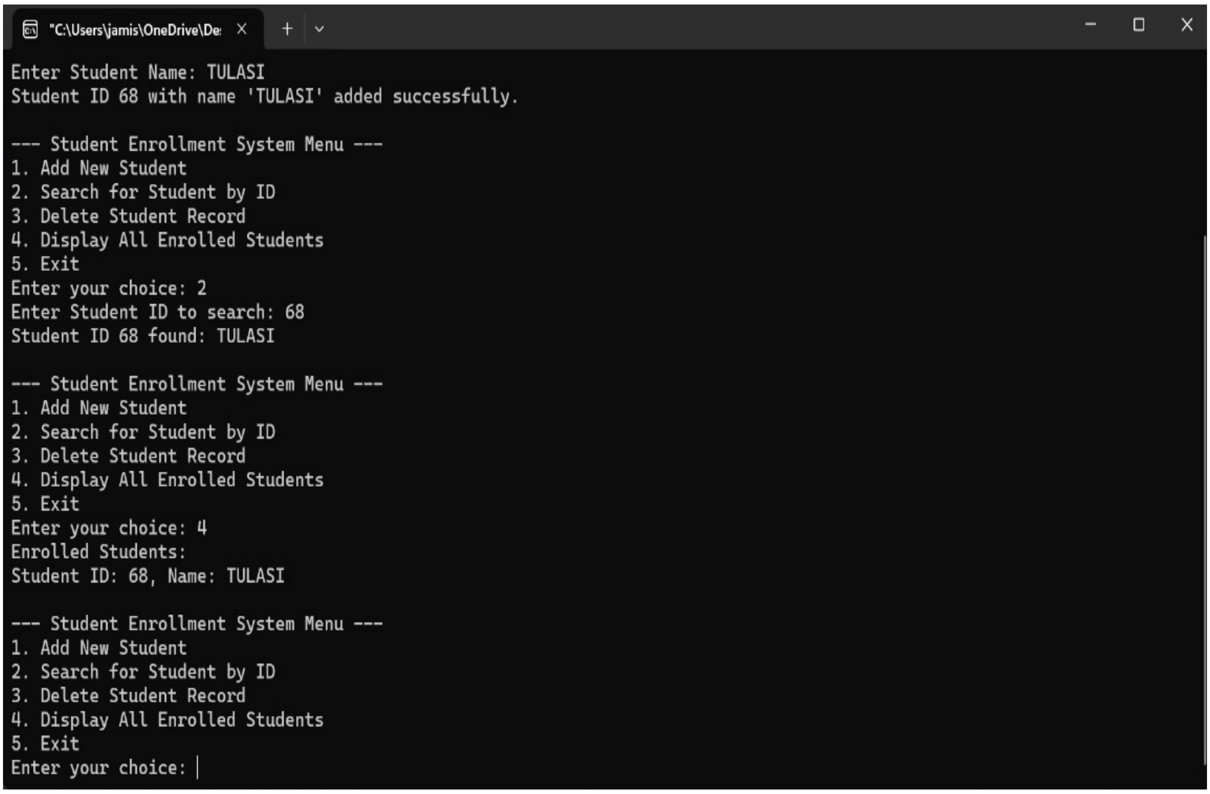
```c
    }
}

// Main function to demonstrate the Student Enrollment System
int main() {
    struct Student* head = NULL; // Head of the linked list
    int choice, studentID;
    char name[100];

    while (1) {
        printf("\n--- Student Enrollment System Menu ---\n");
        printf("1. Add New Student\n");
        printf("2. Search for Student by ID\n");
        printf("3. Delete Student Record\n");
        printf("4. Display All Enrolled Students\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar(); // Consume newline character
        switch (choice) {
            case 1:
                printf("Enter Student ID: ");
                scanf("%d", &studentID);
                getchar(); // Consume newline character
                printf("Enter Student Name: ");
                fgets(name, sizeof(name), stdin);
                name[strcspn(name, "\n")] = '\0'; // Remove newline character
                addStudent(&head, studentID, name);
                break;
```

```
      case 2:
        printf("Enter Student ID to search: ");
        scanf("%d", &studentID);
        struct Student* foundStudent = searchStudent(head, studentID);
        if (foundStudent != NULL) {
          printf("Student ID %d found: %s\n", foundStudent->studentID, foundStudent->name);
        } else {
          printf("Student ID %d not found.\n", studentID);
        }
        break;

      case 3:
        printf("Enter Student ID to delete: ");
        scanf("%d", &studentID);
        deleteStudent(&head, studentID);
        break;
      case 4:
        displayStudents(head);
        break;
      case 5:
        printf("Exiting the program.\n");
        exit(0);
      default:
        printf("Invalid choice! Please try again.\n");
    }
  }

  return 0;
}
```

OUTPUT



```
--- Student Enrollment System Menu ---
1. Add New Student
2. Search for Student by ID
3. Delete Student Record
4. Display All Enrolled Students
5. Exit
Enter your choice: 1
Enter Student ID: 68
Enter Student Name: TULASI
Student ID 68 with name 'TULASI' added successfully.

--- Student Enrollment System Menu ---
1. Add New Student
2. Search for Student by ID
3. Delete Student Record
4. Display All Enrolled Students
5. Exit
Enter your choice: 2
Enter Student ID to search: 68
Student ID 68 found: TULASI

--- Student Enrollment System Menu ---
1. Add New Student
2. Search for Student by ID
3. Delete Student Record
4. Display All Enrolled Students
5. Exit
Enter your choice: 4
Enrolled Students:
```



```
Enter Student Name: TULASI
Student ID 68 with name 'TULASI' added successfully.

--- Student Enrollment System Menu ---
1. Add New Student
2. Search for Student by ID
3. Delete Student Record
4. Display All Enrolled Students
5. Exit
Enter your choice: 2
Enter Student ID to search: 68
Student ID 68 found: TULASI

--- Student Enrollment System Menu ---
1. Add New Student
2. Search for Student by ID
3. Delete Student Record
4. Display All Enrolled Students
5. Exit
Enter your choice: 4
Enrolled Students:
Student ID: 68, Name: TULASI

--- Student Enrollment System Menu ---
1. Add New Student
2. Search for Student by ID
3. Delete Student Record
4. Display All Enrolled Students
5. Exit
Enter your choice: |
```