SELF PROJECT-Student Management System using BST

Main topics used: Binary search Tree,Sorting,Searching.

The **Student Management System (SMS)** is a menu-driven console application developed in **C++** that efficiently manages student data using a **Binary Search Tree (BST)**. Each student record consists of:

- Roll Number (unique key)
- Name
- GPA (Grade Point Average)

The BST is used to maintain the records in a sorted manner based on roll numbers, allowing for efficient **insertion**, **deletion**, **searching**, and **display** operations.

A BST ensures:

- Fast lookup and insertion (O(log n) average-case)
- Naturally ordered data when using **in-order traversal**
- Efficient memory usage for dynamic record handling

This project demonstrates a fundamental use of trees to manage real-world data.

CODE:

```
#include <iostream>
#include <string>
using namespace std;

struct Student {
    int roll;
    string name;
    float gpa;
```

```
    Student* left;
    Student* right;

    Student(int r, string n, float g) {
        roll = r;
        name = n;
        gpa = g;
        left = right = nullptr;
    }
};

class StudentBST {
private:
    Student* root;

    Student* insert(Student* node, int roll, string name, float gpa) {
        if (!node) return new Student(roll, name, gpa);
        if (roll < node->roll)
            node->left = insert(node->left, roll, name, gpa);
        else if (roll > node->roll)
            node->right = insert(node->right, roll, name, gpa);
        else
            cout << "\nDuplicate roll number not allowed." << endl;
        return node;
    }
```

```
    void inorder(Student* node) {
        if (!node) return;
        inorder(node->left);
        cout << "Roll: " << node->roll << ", Name: " << node->name << ", GPA: " << node->gpa <<
endl;
        inorder(node->right);
    }

    Student* search(Student* node, int roll) {
        if (!node || node->roll == roll)
            return node;
        if (roll < node->roll)
            return search(node->left, roll);
        else
            return search(node->right, roll);
    }

    Student* findMin(Student* node) {
        while (node && node->left)
            node = node->left;
        return node;
    }

    Student* remove(Student* node, int roll) {
        if (!node) return nullptr;
        if (roll < node->roll)
```

```
        node->left = remove(node->left, roll);
    else if (roll > node->roll)
        node->right = remove(node->right, roll);
    else {
        // node with one or no child
        if (!node->left) {
            Student* temp = node->right;
            delete node;
            return temp;
        }
        else if (!node->right) {
            Student* temp = node->left;
            delete node;
            return temp;
        }
        // node with two children
        Student* temp = findMin(node->right);
        node->roll = temp->roll;
        node->name = temp->name;
        node->gpa = temp->gpa;
        node->right = remove(node->right, temp->roll);
    }
    return node;
  }

public:
```

```
    StudentBST() {
        root = nullptr;
    }

    void insert(int roll, string name, float gpa) {
        root = insert(root, roll, name, gpa);
    }

    void displayInorder() {
        cout << "\nStudent Records (Sorted by Roll Number):\n";
        inorder(root);
    }

    void searchStudent(int roll) {
        Student* result = search(root, roll);
        if (result)
            cout << "\nFound: Roll: " << result->roll << ", Name: " << result->name << ", GPA: " <<
result->gpa << endl;
        else
            cout << "\nStudent with Roll Number " << roll << " not found." << endl;
    }

    void deleteStudent(int roll) {
        root = remove(root, roll);
    }
};
```

```
int main() {
    StudentBST bst;
    int choice;
    while (true) {
        cout << "\n--- Student Management System ---\n";
        cout << "1. Insert Student\n2. Display All Students\n3. Search Student\n4. Delete Student\n5. Exit\nEnter choice: ";
        cin >> choice;
        if (choice == 1) {
            int roll; string name; float gpa;
            cout << "Enter Roll Number: "; cin >> roll;
            cout << "Enter Name: "; cin >> name;
            cout << "Enter GPA: "; cin >> gpa;
            bst.insert(roll, name, gpa);
        }
        else if (choice == 2) {
            bst.displayInorder();
        }
        else if (choice == 3) {
            int roll;
            cout << "Enter Roll Number to Search: "; cin >> roll;
            bst.searchStudent(roll);
        }
        else if (choice == 4) {
            int roll;
```

```
            cout << "Enter Roll Number to Delete: "; cin >> roll;
            bst.deleteStudent(roll);
        }
        else if (choice == 5) {
            cout << "Exiting...\n";
            break;
        }
        else {
            cout << "Invalid choice. Try again.\n";
        }
    }
    return 0;
}
```