**Data Structures & Algorithms LAB**

(BSCS-F18 Morning & Afternoon)

**Lab # 12**

**Task # 1**

In this lab, you are going to start implementing a class for creating and storing **Binary Search Trees (BST)**. Each node of this BST will store the **roll number**, **name** and **CGPA** of a student.The class definitions will look like:

**class StudentBST;**

**class StudentNode {**

**friend class StudentBST;**

**private:**

**int rollNo; *// Student’s roll number (must be unique)***

**string name; *// Student’s name***

**double cgpa; *// Student’s CGPA***

**StudentNode \*left; *// Pointer to the left subtree of a node***

**StudentNode \*right; *// Pointer to the right subtree of a node***

**};**

**class StudentBST {**

**private:**

**StudentNode \*root; *// Pointer to the root node of the BST***

**public:**

**StudentBST(); *// Default constructor***

**};**

Implement the following two public member functions of the **StudentBST** class:

**bool insert (int rn, string n, double c)**

This function will insert a new student’s record in the **StudentBST**. The 3 arguments of this function are the **roll number**, **name**, and **CGPA** of this new student, respectively. The insertion into the tree will be done based upon the roll number of the student. This function will check whether a student with the same roll number already exists in the tree. If it does not exist, then this function will make a new node for this new student, insert it into the tree at its appropriate location, and return **true**. If a student with the same roll number already exists in the tree, then this function should not make any changes in the tree and should return **false**. This function should NOT display anything on screen.

**bool search (int rn)**

This function will search the **StudentBST** for a student with the given **roll number** (see the parameter). If such a student is found, then this function should display the details (roll number, name, and CGPA) of this student and return **true**. If such a student is not found then this function should display an appropriate message and return **false**.

**Task # 2**

**void inOrder ()**

This function will perform an **in-order** traversal of the **StudentBST** and display the details (roll number, name, and CGPA) of each student. The list of students displayed by this function should be **sorted in increasing order** of roll numbers. This function will be a public member function of the **StudentBST** class. This function will actually call the following helper function to achieve its objective.

**void inOrder (StudentNode\* s) *// private member function of StudentBST class***

This will be a **recursive** function which will perform the in-order traversal on the subtree which is being pointed by **s**. This function will be a **private** member function of the **StudentBST** class.

**Task # 3**

**~StudentBST ()**

This is the destructor for the **StudentBST** class. This function will call the following helper function to achieve its objective.

**void destroy (StudentNode\* s) *// private member function of StudentBST class***

This will be a **recursive** function which will destroy (deallocate) the nodes of the subtree pointed by **s**. This function will be a **private** member function of the **StudentBST** class.

Write a **menu-based** driver function to illustrate the working of all functions of the **StudentBST** class. The menu may look like as shown below:

|  |
| --- |
| **1. Insert a new student**  **2. Search for a student**  **3. See the list of all students (sorted)**  **4. Quit**  **Enter your choice:** |