**Data Structures Lab**

***Session 8***

**Course:** Data Structures (CS2001) **Semester:** Fall 2021

**Instructor:**  **T.A:** N/A

**Note:**

* + - * Lab manual cover following topics

**{Tree, BST, Design and implement classes for binary tree nodes and nodes for general tree, Traverse the tree with the three common orders, Operation such as searches, insertions, and removals on a binary search tree and its applications}**

* Maintain discipline during the lab.
* Just raise your hand if you have any problem.
* Completing all tasks of each lab is compulsory.
* Get your lab checked at the end of the session.

**BST**

**KeyPoint**: A Binary Search Tree (BST) is a binary tree with the following properties:

* The left subtree of a particular node will always contain nodes whose keys are less than that node’s key.
* The right subtree of a particular node will always contain nodes with keys greater than that node’s key. The left and right subtree of a particular node will also, in turn, be binary search trees

class node

{

    public:

    int data;

    node\* left;

    node\* right;

    /\* create Constructor that allocates

    a new node with the given data

    and NULL left and right pointers. \*/

int isBSTUtil(node\* node, int min, int max);

int isBST(node\* node)

{

    return(isBSTUtil(node, INT\_MIN, INT\_MAX));

}

Create a function that returns true if given tree is BST and its values are >=min and <=max.

    int isBSTUtil(node\* node, int min, int max)

{

    /\* an empty tree is BST \*/

    if (node==NULL)

        return 1;

    /\* false if this node violates

    the min/max constraint \*/

    if (node->data < min || node->data > max)

        return 0;

    /\* otherwise check the subtrees recursively,

    tightening the min or max constraint \*/

    return

        isBSTUtil(node->left, min, node->data-1) && // Allow only distinct values

        isBSTUtil(node->right, node->data+1, max); // Allow only distinct values

}

**Task-1:**

Build functionality named autoGrader which will assist DS teacher to check students BST assignments such that if given tree is BST assign 10 points if not then assign 0.

**BST Insertion**

**Sample Code of class Nodes**

Create class Nodes

class Node { private:

int key;

string name;

Node leftChild;

Node rightChild; public:

Node(int key, string name) {

this.key = key;

this.name = name;

}

string toString() {

return cout<<name<< " has the key " <<key<<endl;

} };

**Task-2: Complete the following Code:**

1. **Create class BinaryTree and create a function which add nodes in BST**

class BinaryTree {

private: Node root;

public:

void addNode(int key, string name) {

-----------------------// Create a new Node and initialize it

// If there is no root this becomes root

if (root == NULL) {

----------------------------

} else {

// Set root as the Node we will start with as we traverse the tree

-----------------------------

// Future parent for new Node

Node parent;

while (true) {

// root is the top set the parent node to the root node

--------------------------

// Check if the new node should go on

// the left side of the parent node

Key is compared with that of root. If the key is less than root, it is compared with root’s left child key. If greater, it is compared with the root's right child. Continue this process until the new node is compared with a leaf node and added either on the right or left child depending on its key.

}

1. **Implement main.cpp for the code provided such that a given array is passed to form a BST{ 15, 10, 20, 8, 12, 16, 25 }**