|  |  |
| --- | --- |
| Data Structures & Algorithms Diploma in IT, CSF  Year 2 (2024/25) Semester 4 | Week 13 |
| 2 Hours |
| **Practical 9 – Trees** | |

**Objectives**

At the end of this practical, the students should be able to:

* Understand the concept of Trees
* Implement Binary Search Trees

|  |
| --- |
| **IMPORTANT**   * Upload all your work to Brightspace by the designated timeline stated. |

The specification of Binary Search Tree is given in Appendix A.

1. Implement the method **countNodes()**to count the total number of nodes in the Binary Search Tree.

2. Implement the method **getHeight()**to compute the height of the Binary Search Tree.

3. Implement the method **isBalanced()**to check if the Binary Search Tree is balanced

Appendix A - *Specification of BST (Binary Search Tree)*

|  |
| --- |
| // BST.h - *Specification of BST (Binary Search Tree)*  #pragma once  #include<iostream>  using namespace std;  #include "BinaryNode.h"  class BST  {  private:  BinaryNode\* root; // root of the BST  public:  // constructor  BST();    // search an item in the binary search tree  BinaryNode\* search(ItemType target);  BinaryNode\* search(BinaryNode\* root, ItemType target);    // insert an item to the binary search tree  void insert(ItemType item);  void insert(BinaryNode\* &root, ItemType item);    // delete an item from the binary search tree  void remove(ItemType target);  void remove(BinaryNode\* &root, ItemType target);    // traverse the binary search tree in inorder  void inorder();  void inorder(BinaryNode\* t);    // traverse the binary search tree in preorder  void preorder();  void preorder(BinaryNode\* t);    // traverse the binary search tree in postorder  void postorder();  void postorder(BinaryNode\* t);    // check if the binary search tree is empty  bool isEmpty();    // ----------------- to be implemented ------------------  // count the number of nodes in the binary search tree  int countNodes();  int countNodes(BinaryNode\* t);  // compute the height of the binary search tree  int getHeight();  int getHeight(BinaryNode\* t);  // check if the binary search tree is balanced  bool isBalanced();  bool isBalanced(BinaryNode\* t);  }; |

BinaryNode.h

|  |
| --- |
| #pragma once  typedef int ItemType;  struct BinaryNode  {  ItemType item; // data item  BinaryNode\* left; // pointer pointing to left subtree  BinaryNode\* right; // pointer pointing to right subtree  }; |