CSCE4263 – Advanced Data Structures
Fall 2025

Quiz 1
Binary Tree

Date: Aug. 26, 2025

Time: 30 minutes

Instructions:

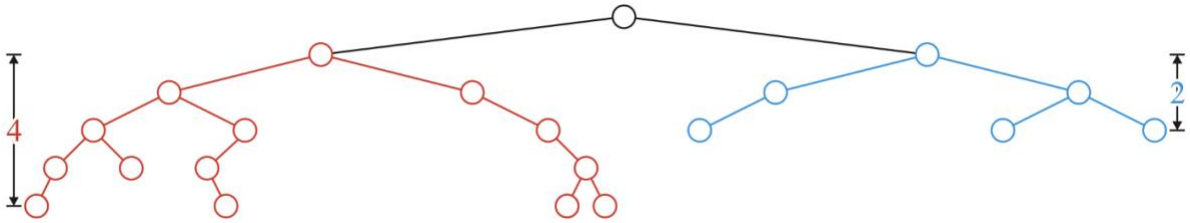
- **Written Format & Template:** Students can use either Google Doc or MS Word
- Write your full name, email address and student ID in the report.

```
class Binary_node {
   protected:
      int node_value;
      Binary_node *p_left_tree;
      Binary_node *p_right_tree;
   public:
      Binary_node( Type const & );

      int value() const;
      Binary_node *left() const;
      Binary_node *right() const;
      bool is_leaf() const;
      int size() const;
      int height() const;
      void clear();
 }

bool Binary_node::is_leaf() const {
   // Return True if the current node is the leaf
   return (left() == nullptr) && (right() == nullptr);
}

int Binary_node::height() const {
   // Return the height of the current node
   if ( left() == nullptr ) {
            return ( right() == nullptr ) ? 0 : 1 + right()->height();
   } else {
            return ( right() == nullptr ) ?
            1 + left()->height() :
            1 + left()->height() + right()->height();
   }
}
```


4
2

Question 1: Write the function to count the number of leaves in a binary tree. (30 points)

```
int count_leaves(Binary_node *root) {
    if (root == nullptr) // If this is an empty tree, return 0
        return 0;
    if (root->is_leaf()) // If this is a leaf, return 1. Leaf does not have children
        return 1;
    // Total leaves = #leaves in left subtree + #leaves in right subtree
    return count_leaves(root->p_left_tree) + count_leaves(root->p_right_tree);
}
// int number_of_leaves = count_leaves(tree_root);
```

Question 2: Write the functions to delete the highest leaf in a binary tree. If there is more than one, you can delete any of them. (70 points)

```
Binary_node* delete_highest_leaf(Binary_node *root) {
    if (root == nullptr) // If this is an empty tree, return a null pointer
        return nullptr;
    if (root->is_leaf()) {
        // We found a highest leaf
        // Delete it and return a null pointer
        delete root;
        return nullptr;
    }
    // Up to this point, root must have children
    if (root->p_left_tree == nullptr) {
        // If the root does not have left subtree
        // The highest leaf must be in right subtree
        // Delete the highest leaf in the right subtree and adjust the pointer
        root->p_right_tree = delete_highest_leaf(root->p_right_tree);
    } else if (root->p_right_tree == nullptr) {
        // If the root does not have right subtree
        // The highest leaf must be in left subtree
        // Delete the highest leaf in the left subtree and adjust the pointer
        root->p_left_tree = delete_highest_leaf(root->p_left_tree);
    } else if (root->p_left_tree->height() > root->p_right_tree->height()) {
        // The root has both left and right subtree
        // If the height of left subtree is higher than the right subtree
        // The highest leaf will be the left subtree
```

```
        // Delete the highest leaf in the left subtree and adjust the pointer
        root->p_left_tree = delete_highest_leaf(root->p_left_tree);
    } else {
        // Othewise,
        // The highest leaf must be in right subtree
        // Delete the highest leaf in the right subtree and adjust the pointer
        root->p_right_tree = delete_highest_leaf(root->p_right_tree);
    }
}
```