CSCE4263/5183 – Advanced Data Structures
Fall 2025

Quiz 2
AVL Trees

Date: Sep. 30, 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.
- Submission (MS Word or pdf) via BlackBoard

Question 1: (20 pts)

When is the binary search tree AVL balanced?

Answer: A binary search tree is an AVL if and only if for every single node in the binary search tree, the height difference between its left subtree and right subtree is no more than 1 (i.e., -1 <= left height – right height <= 1)

Question 2: (30 pts)

Given a binary search tree as in Fig. 1

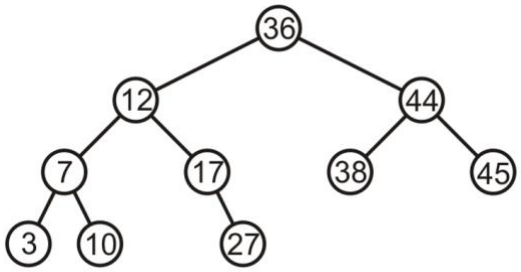


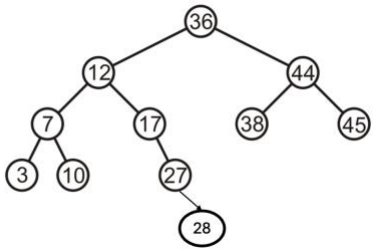
Figure 1. Binary search tree

a. Is it AVL balanced?

Answer: Yes, this AVL tree is balanced since there is no node whose absolute balanced factor is greater than 1.

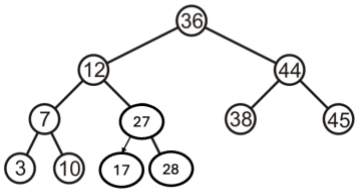
b. Insert note 28 into this tree, show the new tree.

Answer:



c. Is the new tree AVL balanced? If not, fix it.

Answer: The tree not is not balanced since the absolute balance factor of node 17 is greater than 1.



Question 3: (50 pts)

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

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

AVL_node is inherited from Binary_node

```
bool AVL_node::insert(int & obj ) {
   if (obj < value() ) {
      if (left() == nullptr ) {
         p_left_tree = new AVL_node( obj );
         tree_height = 1;
      } else if (left()->insert(obj, p_left_tree ) ) {
         tree_height = max(height(), 1 + left()->height() );
         return true;
      } else {
         return false;
      }
   } else if (obj > value() ) {
      // To be completed

   } else {
```

```
        return false;
    }
}
```

Complete the section [// To be complete] above.

```
bool AVL_node::insert(int & obj ) {
    if (obj < value() ) {
        if (left() == nullptr ) {
            p_left_tree = new AVL_node( obj );
            tree_height = 1;
        } else if ( left()->insert(obj) ) {
            tree_height = max( height(), 1 + left()->height() );
            return true;
        } else {
            return false;
        }
    } else if (obj > value() ) {
        // This is similar to inserting into the left subtree
        if (right() == nullptr) {
            p_right_tree = new AVL_node( obj );
            tree_height = 1;
        } else if ( right()->insert(obj) ) {
            tree_height = max( height(), 1 + right()->height() );
            return true;
        } else {
            return false;
        }
    } else {
        return false;
    }
    return true; // Return True when we insert successfully
}
```