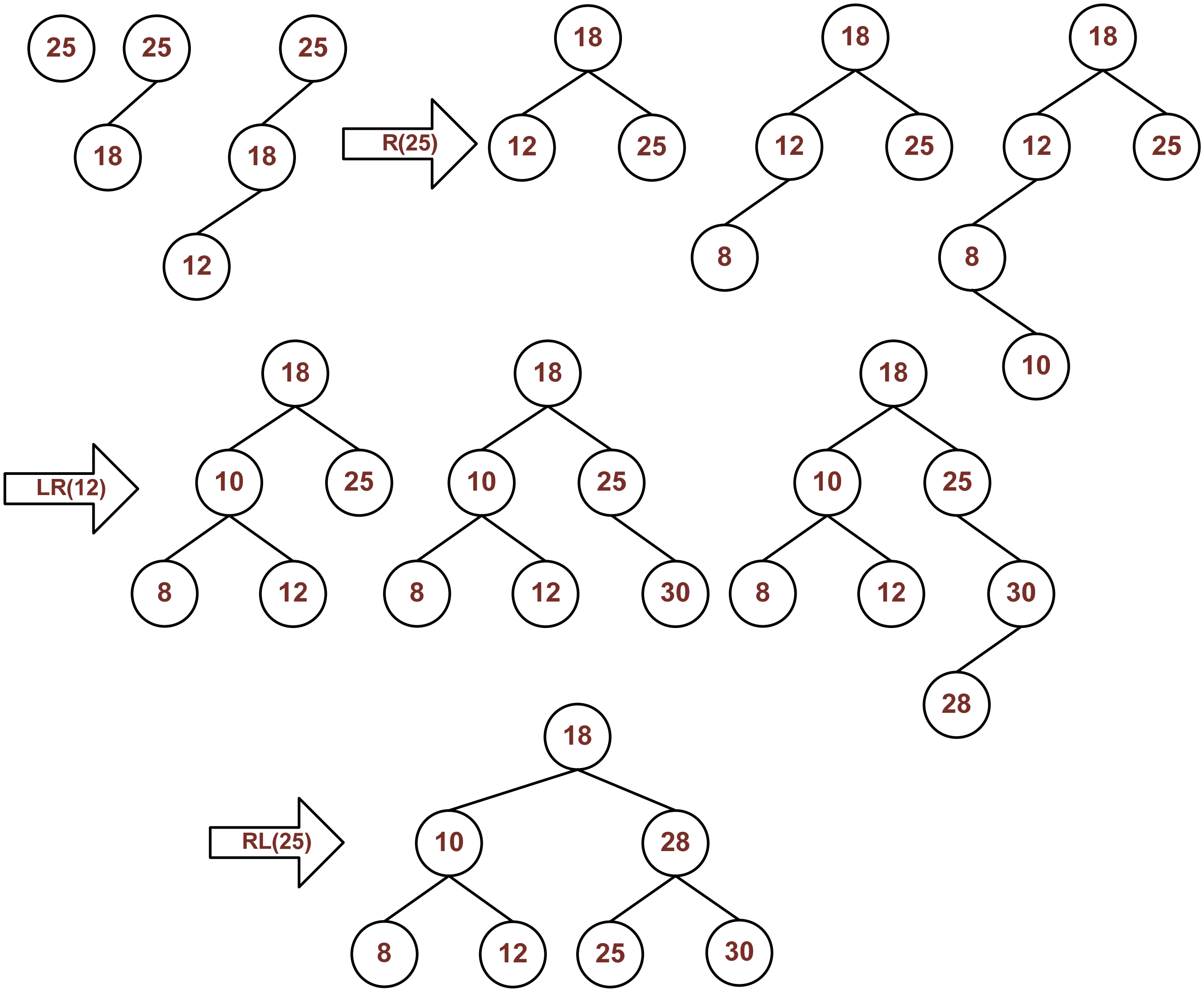
Review Activity 13 Solutions

AVL Trees, Tree-Based Algorithms

1) Given an empty AVL tree, insert the following values into the tree: `25 18 12 8 10 30 28`
   Show the rotations used in deriving your solution, and show the final tree structure.



2) Design a recursive function "**bool check_if_BST(BinaryTreeNode* root)**" that takes as input the root node of a tree; each node stores "**int value**". The function outputs **true** if the given tree is a valid binary search tree (BST), and **false** otherwise.

In a valid BST instance, the BST property holds at every node; also, empty tree is a BST. You may assume that "**int min_value(BinaryTreeNode*)**" and "**int max_value(BinaryTreeNode*)**" are available as helper functions to output minimum and maximum values in a subtree, respectively.

```
bool check_if_BST(BinaryTreeNode* root) {
// write your function below
     if (!root)
          return true; // empty tree is a BST

     if (root->left && max_value(root->left) >= root->value)
          return false; // no duplicates in left subtree

     if (root->right && min_value(root->right) < root->value)
          return false; // allow duplicates in right subtree

     if (!check_if_BST(root->left) || !check_if_BST(root->right))
          return false; // recursively check left and right subtrees

     return true;
}
```