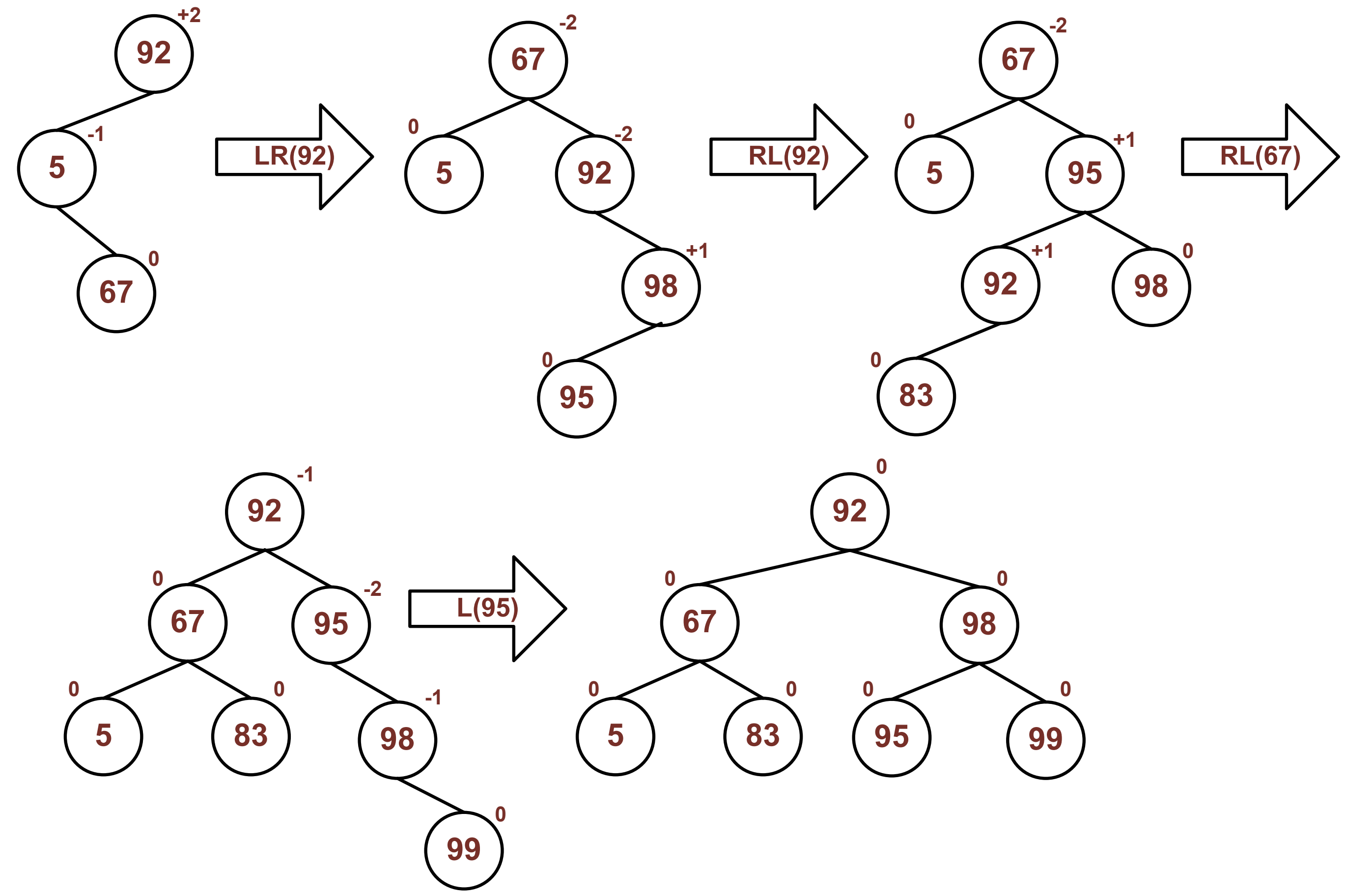
Review Activity 15 Solutions

AVL Trees, Tree-Based Algorithms: Additional Practice

1) Given an empty AVL tree, insert the following integer values into the tree: 92 5 67 98 95 83 99. Show the rotations used in deriving your solution, and write the avlBalance value for each node before rotations. Check the balancing in a bottom-up manner, by finding the first node starting from the bottom for which the |avlBalance| > 1.



2) Write the function `performRRotation` that performs the right AVL rotation at pNode. `pNodeAddress` is the address of a pointer to the pNode, such the parent node's left or right child, or the root node pointer. Update `pNodeAddress` value when appropriate.

To get you started, we have provided qNode as `pNode->left`. Use this pointer in your implementation. Adequately document your code.

```
void AVLTree::performRRotation(BSTNode* pNode, BSTNode** pNodeAddress) {
      BSTNode* qNode = pNode->left;

      // implement your code below

      // adjust pNode and qNode
      pNode->left = qNode->right;
      qNode->right = pNode;

      // adjust the parent node
      *pNodeAddress = qNode;
```

3) Write the function `performRLRotation` that performs the right-left AVL rotation at pNode. The function first applies the right rotation at `pNode's` right child, and then it applies the left rotation at pNode. `pNodeAddress` is the address of a pointer to the pNode, such the parent node's left or right child, or the root node pointer. Update `pNodeAddress` value when appropriate.

To get you started, we have provided qNode as `pNode->right` and rNode as `qNode->left`. Use these pointers in your implementation. Adequately document your code.

```
void AVLTree::performRLRotation(BSTNode* pNode, BSTNode** pNodeAddress) {
      BSTNode* qNode = pNode->right;
      BSTNode* rNode = qNode->left;

      // implement your code below

      // adjust pNode and qNode
      pNode->right = rNode->left;
      qNode->left = rNode->right;

      // adjust rNode
      rNode->left = pNode;
      rNode->right = qNode;

      // adjust the parent node
      *pNodeAddress = rNode;
```