# Midterm 1 Solutions

**1. (10 Points)**

Show the AVL tree that results after **each** of the integer keys $9, 27, 50, 15, 2, 21,$ and $36$ are inserted, in that order, into an initially empty AVL tree. Clearly show the tree that results after each insertion, and make clear any rotations that must be performed.

**Solution:**
See figure 1.

**2. (10 Points)**

Show the red-black tree that results after each of the integer keys $21, 32, 64, 75,$ and $15$ are inserted, in that order, into an initially empty red-black tree. Clearly show the tree that results after **each** insertion (indicating the color of each node), and make clear any rotations that must be performed.

**Solution:**
See Figure 2 for bottom-up approach, and Figure 3 for top-down approach.

**3. (12 Points)**

Indicate for each of the following statements if it is true or false. You must justify your answers to get credit.

(a) The subtree of the root of a red-black tre is always itself a red-black tree. (Here, the definition of red-black tree is as I have given in class and as described in the textbook.)

(b) The sibling of a null child reference in a red-black tree is either another null child reference or a red node.

(c) The worst case time complexity of the insert operation into an AVL tree is $O(\log n)$, where $n$ is the number of nodes in the tree.

**Solution:**

(a) FALSE. The root of a red-black must be black, by definition. It is possible for the child of the root of a red-black tree to be red. Therefore, it is possible for the subtree of the root of a red-black tree to have a red root, meaning that it can not be a red-black tree. So, the statement is false.

(b) TRUE. Let $x$ represent the parent of the null reference, and without loss of generality, suppose x.right is the null reference. Suppose x.left refers to a black node. But then, the number of black nodes on the path from the root to the x.right null reference must be less than the number of black nodes from the root to all nodes in the subtree rooted at x.left. This violates the definition of red-black tree. So, x.left must be a null reference or a red node.

(c) TRUE. The work of all AVL tree operations is $O(h)$ where $h$ is the height of the tree. AVL rotations ensure that $h$ is $O(\log n)$. Therefore , insertion must be $O(\log n)$.

**4. (10 Points)**

Suppose you have an AVLNode class that stores integers:

```
public class AVLNode {
    public int item;
    public AVLNode left;
    public AVLNode right;
    public AVLNode (int i, AVLNode l, AVLNode r)
        item = $i$; left = $l$; right = $r$; ;
}
```

Write a complete method that takes a height $h$, and returns a reference to the root of an AVL tree of height $h$ that contains the minimum number of nodes. The integers stored in the nodes (the item instance variables) should satisfy the binary search tree property, and they should all be distinct (but they do not have to be consecutive). You can define helper methods and/or classes if you wish.

**Solution:**

An example solution is as follows.

```
public AVLNode minNodeAVLTree (int $h$) {
    return minNodeAVLTree($h, 1$);
}

private AVLNode minNodeAVLTree (int $h$, int minVal)
{
    if ($h == -1$) {
        return $null$;
    } else {
        /* Numbers do not have to be consecutive. At most $2^h - 1$ nodes on the left subtree. */
        int rootVal = minVal + (Math.pow($2, h$)$-1$);
        AVLNode left = minNodeAVLTree ($h - 1$, minVal);
        AVLNode right = minNodeAVLTree ($h - 2$, rootVal+1);
        return new AVLNode (rootVal, left, right);
    }
}
```

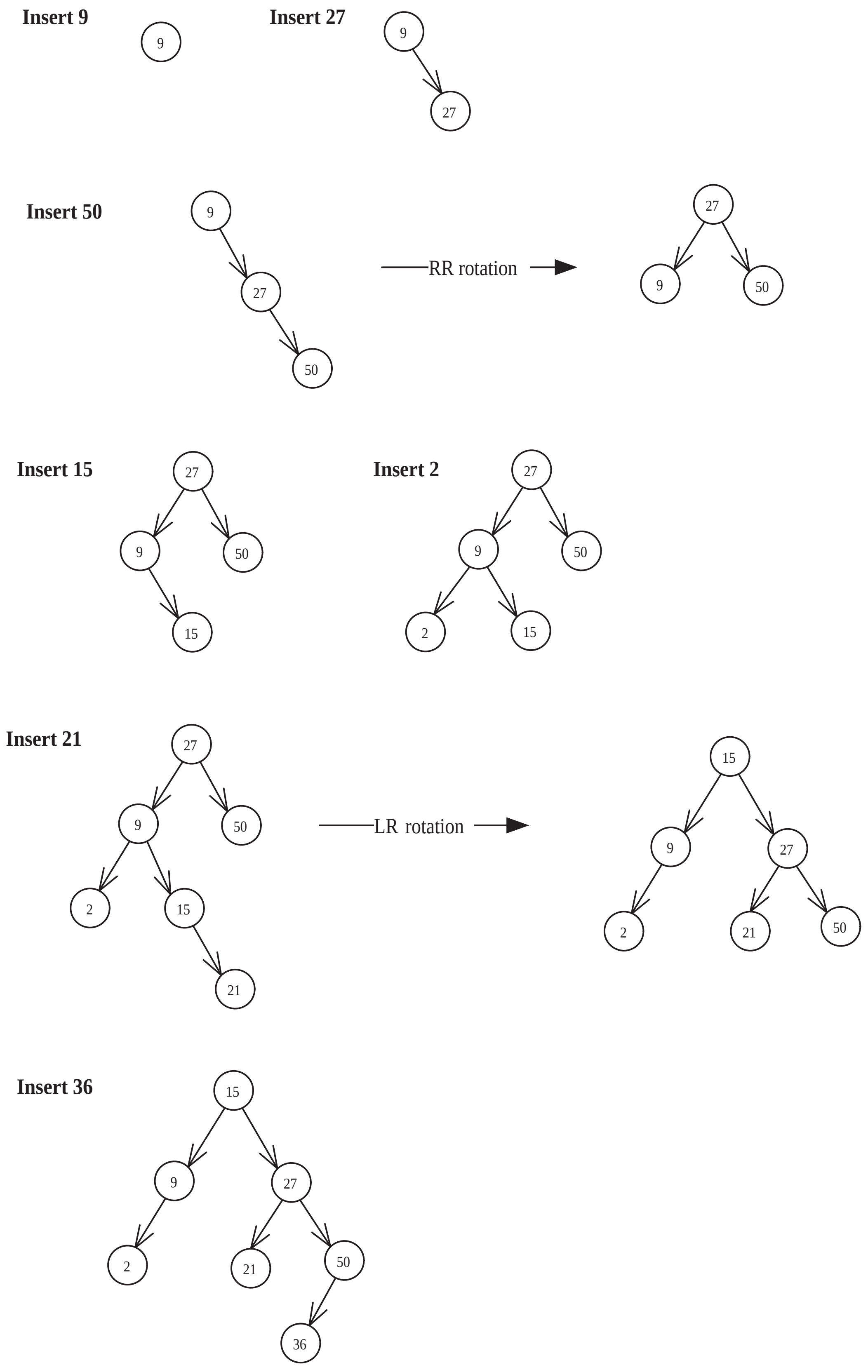

Figure 1: Problem 1. Insertion in an AVL tree

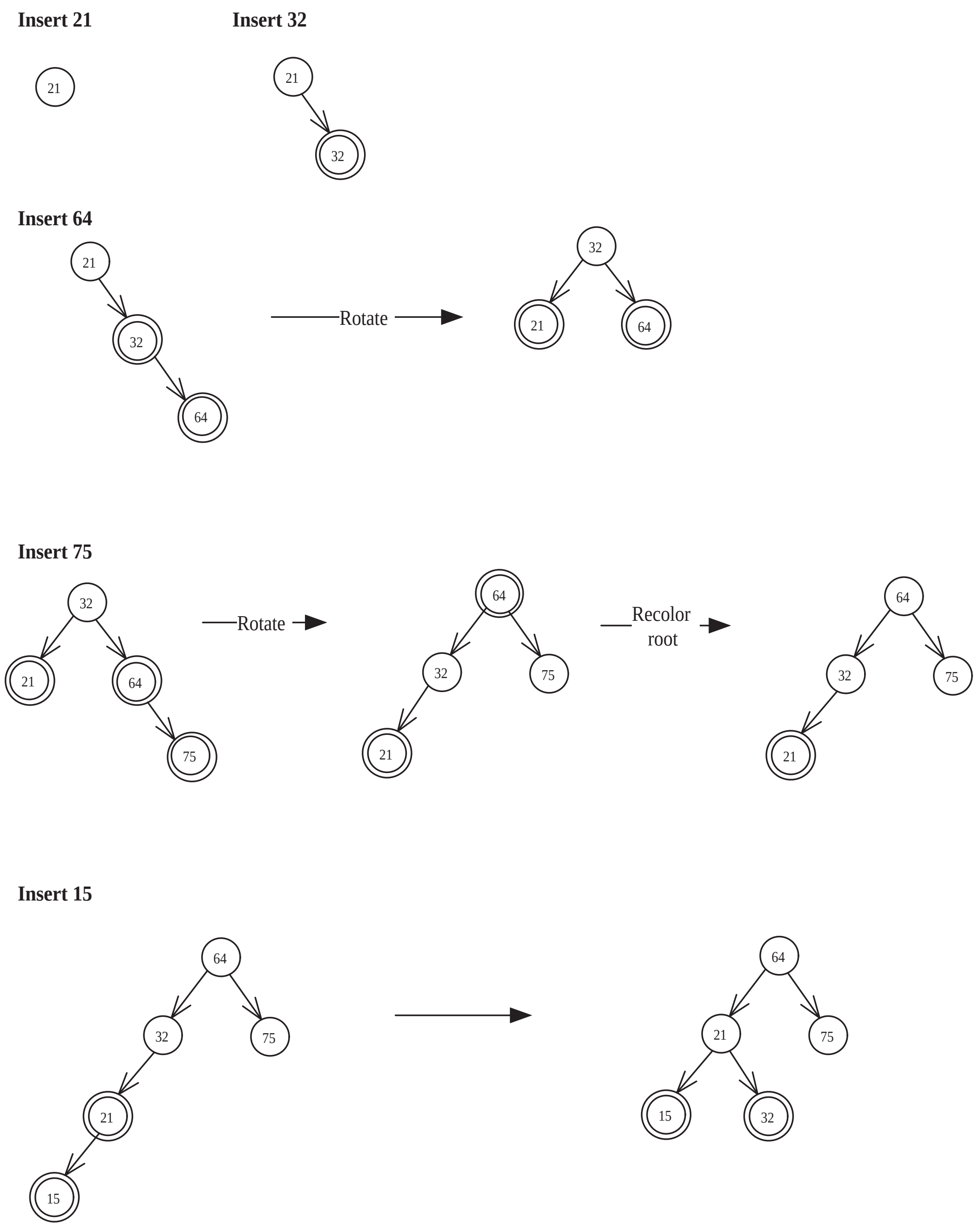

Figure 2: Problem 2. Insertion in a Red-black tree. Bottom-up approach

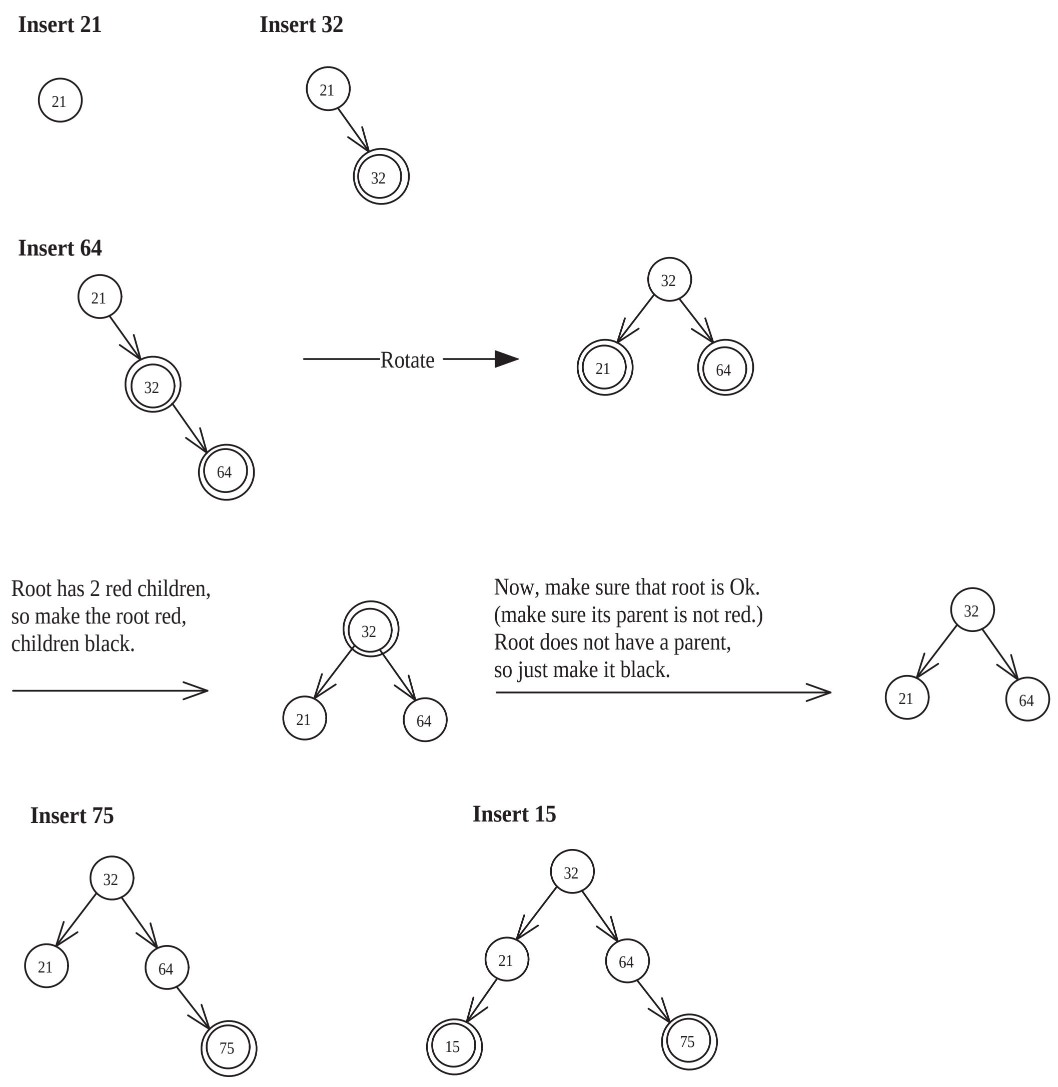

Figure 3: Problem 2. Insertion in a Red-black tree. Top-down approach.