HOMEWORK-7
Total Points: 65

1. [10 Points] If n_1, n_2, n_3, .., n_k are the common ancestors of given two nodes a and b and length(n_i, a) is the length of the path from n_i to a, then the least common ancestor (LCA) of a and b is n_i such that length(n_i, a) <= length(n_j, a) for all 1 <= j <= k. Give an algorithm to find the LCA of two given nodes in a BST of integers. You can assume that all the nodes contain distinct values.
   The prototype of the lca routine is as follows:
   ```
   lca(root, v1, v2)
   root : is the root of the BST
   v1: the integer value stored in the first node
   v2: the integer value stored in the second node
   the return value is the integer value stored in the LCA
   ```

2. [15 Points] Give an algorithm to find the LCA (as discussed in the previous question) of two given nodes in a binary tree of integers. You can assume that all the nodes contain distinct values. You can add additional fields (e.g., a pointer to the parent, or a status field, etc.) in a node of the binary tree to compute LCA efficiently.
   The prototype of the lca routine is as follows:
   ```
   lca(root, v1, v2)
   root : is the root of the binary tree
   v1: the integer value stored in the first node
   v2: the integer value stored in the second node
   the return value is the integer value stored in the LCA
   ```

3. [5 points] What is the minimum number of nodes in an AVL tree of height 10? Show all intermediate computations.

4. [10 Points] Insert 10, 20, 30, 24, 27, 22, 6, 4, 23, and 1 one-by-one in an AVL tree in the given order starting from an empty tree. Show all intermediate steps. Before each insertion, if applicable, clearly mention which rotation is required.

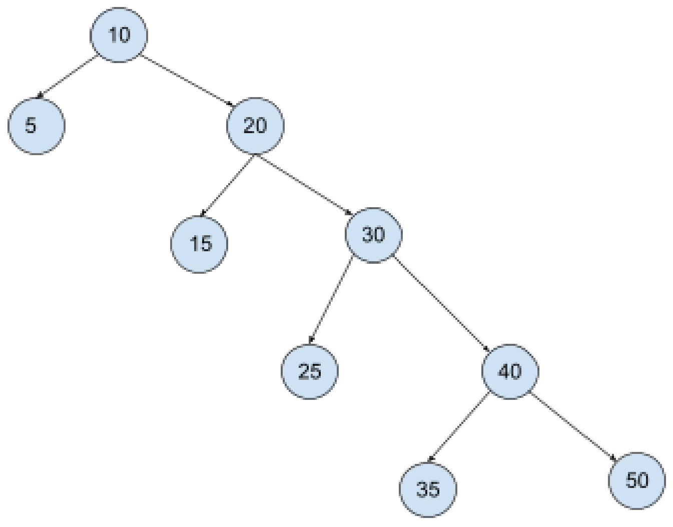
5. [25 Points] In a BST, the number of operations required to search a key at depth d is directly proportional to d. Algorithm QuickSearch tries to reduce the number of operations when the same keys are frequently searched. During a search operation, if n is the node that contains the key, and n is not root, QuickSearch reduces the depth of n by one before returning it to the caller. The depths of other nodes may also change. QuickSearch doesn't violate the BST property during the reordering. For a successful search operation of a node at depth d, the time complexity should be O(d). In the following example, initially, the depth of 35 is 4. Let's say the application tries to search the following keys in the given order: 35, 35, 35, 15, 5.

After the first search operation, the depth of 35 will be 3.
After the second search operation, the depth of 35 will be 2.
After the third search operation, the depth of 35 will be 1.
After the fourth search operation, the depth of 15 will be either d-1 if d > 0 or zero, where d is the original depth of 15 before the search.
After the fourth search operation, the depth of 5 will be either d-1 if d > 0 or zero, where d is the original depth of 5 before the search.
The depth of other nodes may also change during a search operation.

Give an algorithm for QuickSearch. QuickSearch returns the address of the node that contains the key and the root (possibly new) of the BST. If the key is not present, QuickSearch returns NULL and the original root of the BST.

You are not allowed to change the type of a BST node. The type of a BST node is:

```
struct Tree {
  int val;
  struct Tree *left;
  struct Tree *right;
};
```


10
5
20
15
30
25
40
35
50