Assignment 4

Leila Erbay

ID: 260672158

**Question 2:**

1. To obtain a tree with the minimal height, one should find the median of the keys and insert based on that value so that the smaller values are on the left and larger values are on the right thus the tree will be balanced.
2. The worst-case running time of the find, insert, and remove operations based on (a) would be O(log n).
3. To obtain a tree with maximal height, the keys should be inserted from smallest to largest for all the nodes to be inserted on the left or largest to smallest values for every node to be inserted on the right. The tree in both cases will be unbalanced.
4. The worst-case running time of the find, insert, and remove operations based on (c) would be O(n).

**Question 3:**

1. weirdPreOrder(treeNode n):

7, 9, 9, 7, 8, 6, 2, 1, 2, 3

1. weirdPostOrder(treeNode n):

9, 9, 7, 8, 6, 2, 1, 2, 3, 7

1. queueTraversal(treeNode n):

7, 3, 9, 2, 6, 8, 9, 1, 2, 7

1. stackTraversal(treeNode n):

7, 3, 2, 1, 2, 6, 9, 8, 7, 9

equivalent to preorderTraversal implementation of Depth-First Search

**Question 4:**

Algo: IsIsomorph(Tree A, Tree B)

In: Two trees to compare

Out: Boolean telling if the two trees are isomorph

if (A.getValue() == null) && (B.getValue() == null) && ((A.getLeftChild() == null) && (A.getRightChild() == null) && (B.getLeftChild() == null) && (B.getRightChild() == null)){

return true

}

if ((A.getLeftChild() == B.getRightChild()) || (A.getLeftChild() == B.getLeftChild())) {

return A.getLeftChild().IsIsomorph(A.getLeftChild()) && A.getRightChild().IsIsomorph(B.getLeftChild());

}

if ((A.getLeftChild() == B.getRightChild()) || A.getRightChild() == B.getRightChild()){

return A.getLeftChild().IsIsomoprh(B.getLeftChild()) && A.getRightChild().IsIsomorph(B.getRightChild());

}

return false;

}