Quiz 4 (Oct 11)

By taking this quiz, you agree to adhere to the honor code of the class.

Name: netid:

Write your name and netid on **both** sides of the paper. Write your solution **first on this side**. If space is not enough, write to the other side. You can ask for extra paper if necessary.

Name: netid:

Create a method `List<int> allHeights(Position<E> root)` that returns a list of all the heights of the nodes of the binary tree in a postorder traversal. This implementation should run in O(n) time, i.e. do not do a preorder traversal just modifying list.addLast(height(p)). Instead, use height as an auxiliary variable in a recursive method.

Signatures

```
public class LinkedBinaryTree<E> implements BinaryTree<E> {
        private class Node implements Position<E>{
                private E element;
                private Position<E> left;
                private Position<E> right;
                private Node parent;

                public Node(E element, Node parent) {
                        this.element = element;
                        this.parent = parent;
                }

                public E getElement(){return element;}
                public Node getParent() {return parent;}
                public Position<E> getLeft() {return left;}
                public Position<E> getRight() {return right;}
        }
```

Reference solution

```java
public List<Integer> allHeights(Position<E> root) {
    List<Integer> result = new ArrayList<>();
    helper(root, result);
    return result;
}

private int helper(Positoin<E> node, List<Integer> result) {
    if (node == null) {
        return -1;
    }
    int left_height = helper((Node)node.left, result);
    int right_height = helper((Node)node.right, result);
    int height = Math.max(left_height, right_height) + 1;
    result.add(height);
    return height;
}
```