Assignment 8

Name: DONGWOOK LEE

Problem 8.2 *Linked Lists & Rooted Trees*

(c)

StackUsingLinkedList.py | BinarySearch2LinkedList.py | LinkedList2BinarySearch.py

```
class Tree:
    def __init__(self, key):
        self.root = None
        self.key = key
        self.parent = None
        self.left = None
        self.right = None


def tree_insert(tree, z):
    y = None
    x = tree.root
    while x is not None:
        y = x
        if z.key < x.key:
            x = x.left
        else:
            x = x.right
    z.parent = y
    if y is None:
        tree.root = z
    elif z.key < y.key:
        y.left = z
    else:
        y.right = z
```

```
L = [2, 3, 4, 6, 7, 9, 13, 15, 17, 18, 20]
T = Tree(None)

for i in range((len(L)//2), len(L)):
    tree_insert(T, Tree(L[i]))

for i in range(0, len(L)//2):
    tree_insert(T, Tree(L[i]))
```

Search Time Complexity: O(h)

In Linked List, Search Time Complexity is O(n) where n is the number of elements in the list

However, in Binary Search Tree, we can branch out the elements in two ways,

which let value of h to be lower than n

Thus, Search Time Complexity of this Algorithm is indeed lower than that of sorted Linked List;

- $O(n) > O(h)$ where $n > h$