Lab6 Assignment – AVL Trees

Do these problems in order. While implementing/debugging, hard-code the program's input in your Python file. Once your code is working you can prompt the user for input.

Problem 1: AVL Tree operations.

Implement the Ordered Dictionary ADT operations using **AVL Trees**. You will have implement the following methods:

`search, minimum, maximum, successor, predecessor, insert, delete`.

You can re-use code from the previous assignment for the TreeNode class.

To test your code, you can create a BST in the main (or in a separate function) and run the operations on that tree. To print a BST, you can define & use a method **`preorder-traverse`** that prints the pre-order traversal of the keys in the BST.

Problem 2: AVL Tree vs BST.

Design an experiment to compare the performance of the operations in a usual BST and an AVL Tree. For e.g. you can insert a large number (say 10000) keys into each data structure and measure the time taken for the two cases. What about the runtime of the other operations? For start, you can fill in the following table by computing the runtime for each entry of the table.

	Insert n=1000	Insert n=10000	Insert n=100000	Search n=1000	Search n=10000	Search n=100000	Delete n=1000	Delete n=10000
BST								
AVL Tree								

Can you observe any relation between the runtime and the order of the keys inserted?

You can modify your call to the main function as follows to measure the runtime of your program:

```
start_time = time.time()
main()
print("--- %s seconds ---" % (time.time() - start_time))
```

You will need to import Python's time module by adding the following line at the start of your program:

```
import time
```