HW 2 - Binary Search Trees

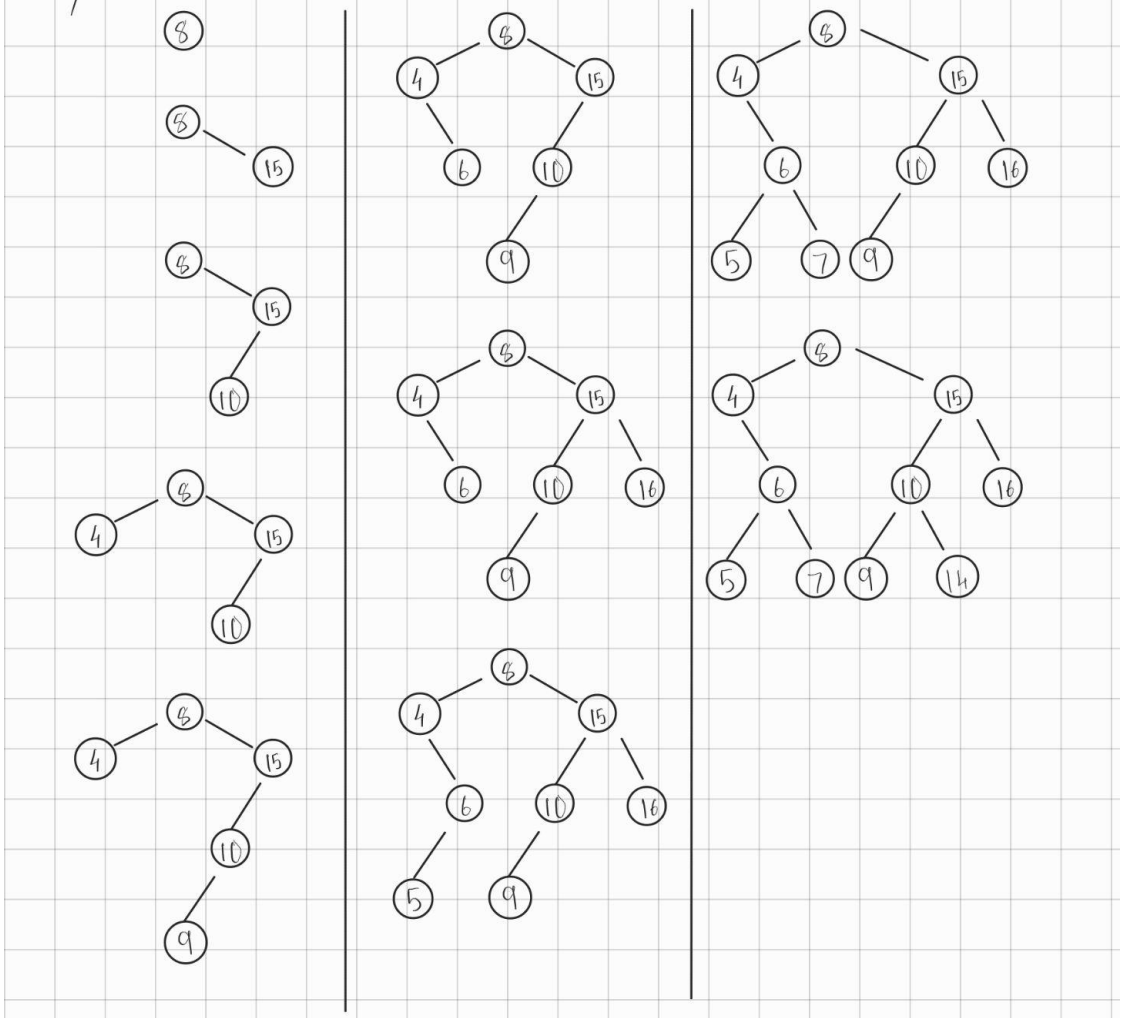
CS 202

Section # 2

Ghulam Ahmed (22101001)

Question 1

a)



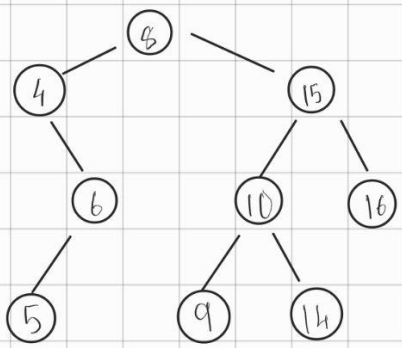
b) PreOrder: 8, 4, 6, 5, 7, 15, 10, 9, 14, 16

InOrder: 4, 5, 6, 7, 8, 9, 10, 14, 15, 16

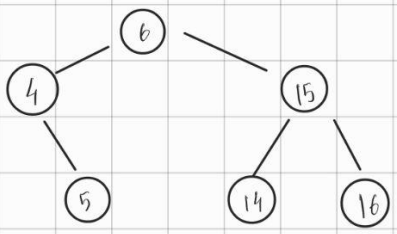
PostOrder: 5, 7, 6, 4, 9, 14, 10, 16, 15, 8

c) Delete

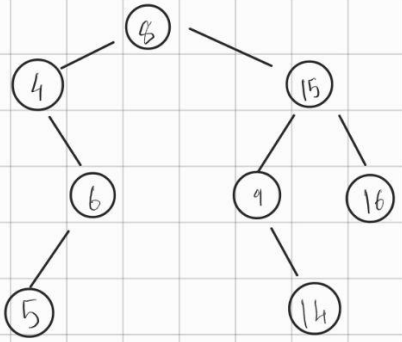
7:



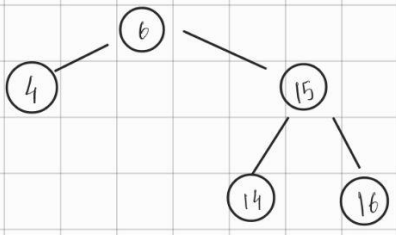
9:



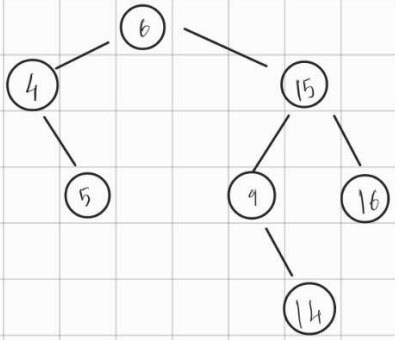
10:



5:



8:



d) Write a recursive pseudocode implementation for finding the minimum element in a binary search tree.

```
function findMin(node)
        if node is null
                return null
        else if node.left is null
                return node.item
        else
                return findMin(node.left)
```

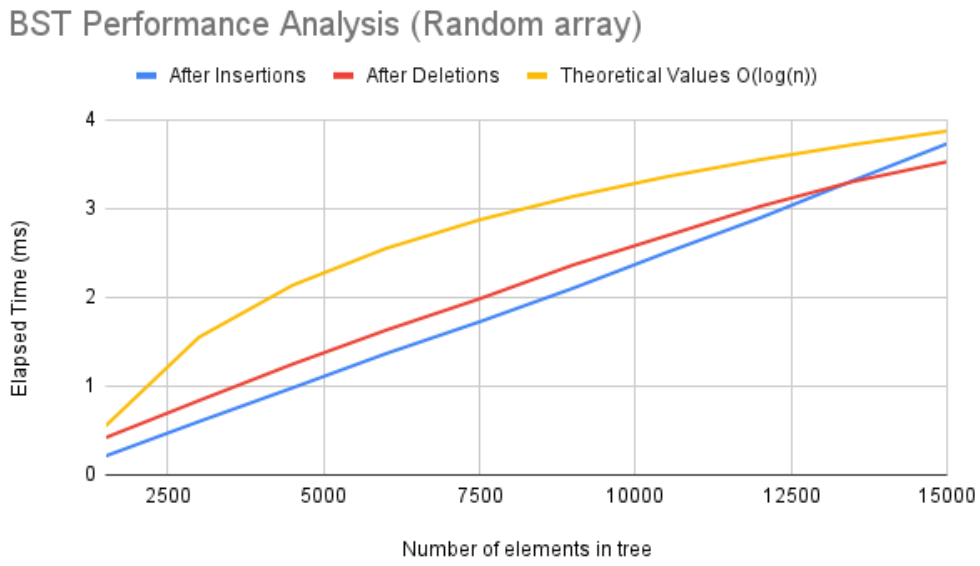
e) What is the maximum and minimum height of a binary search tree that contains n items ?

Minimum height: The height of the tree is the minimum, in best case, and it is equal to **log $_2$ (n)**

Maximum height: The height of the tree is maximum, in worst case, and it is equal to n

Question 3

● Interpret and compare your empirical results with the theoretical ones. Explain any differences between the empirical and theoretical results, if any.



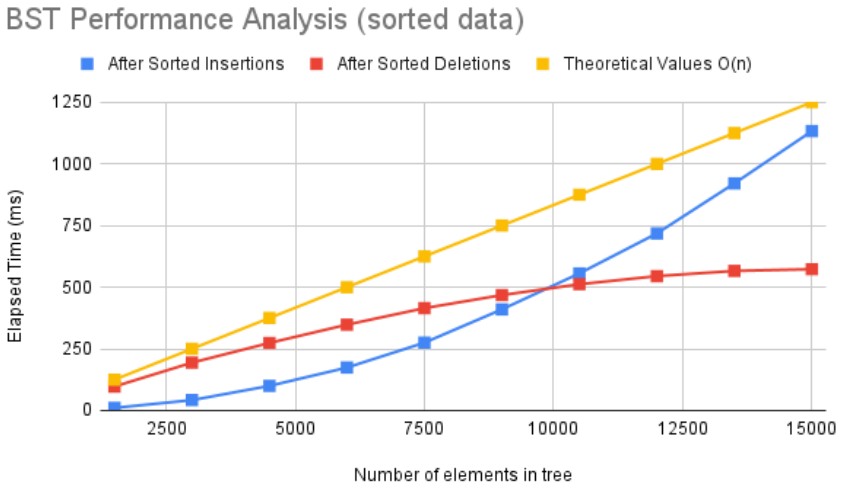

```
Part e - Time Analysis of  Binary Search Tree - part 1
-------------------------------------------------------
Tree Size           Time elapsed
-------------------------------------------------------
1500                0.212   ms
3000                0.601   ms
4500                0.98    ms
6000                1.366   ms
7500                1.724   ms
9000                2.103   ms
10500               2.505   ms
12000               2.894   ms
13500               3.317   ms
15000               3.73    ms

Part e - Time Analysis of  Binary Search Tree - part 2
-------------------------------------------------------
Tree Size           Time elapsed
-------------------------------------------------------
1500                0.419   ms
3000                0.837   ms
4500                1.247   ms
6000                1.63    ms
7500                1.983   ms
9000                2.364   ms
10500               2.693   ms
12000               3.024   ms
13500               3.305   ms
15000               3.527   ms
```

The time complexity of insertion and deletion into a binary search tree is O(log(n)) in the average case (i.e. Inserting random items). The empirical results prove this in the above chart where the theoretical line closely follows the

empirical result lines. The theoretical line is obtained by taking the logarithm of the number of elements in the tree.

● How would the time complexity of your program change if you inserted sorted numbers into it instead of randomly generated numbers?




```
Part e - Time Analysis of  Binary Search Tree - part 1
-------------------------------------------------------
Tree Size           Time elapsed
-------------------------------------------------------
1500                10.631  ms
3000                42.719  ms
4500                99.766  ms
6000                174.586 ms
7500                275.337 ms
9000                410.266 ms
10500               556.764 ms
12000               718.657 ms
13500               921.932 ms
15000               1133.34 ms

Part e - Time Analysis of  Binary Search Tree - part 2
-------------------------------------------------------
Tree Size           Time elapsed
-------------------------------------------------------
1500                97.152  ms
3000                194.19  ms
4500                274.144 ms
6000                348.639 ms
7500                415.436 ms
9000                468.923 ms
10500               512.076 ms
12000               545.932 ms
13500               566.962 ms
15000               573.574 ms
```

If we insert sorted numbers into the tree instead of randomly generated numbers, the worst-case time complexity will change from O(log n) to O(n) for both insertion and deletion operations. This is because the tree becomes unbalanced and starts acting like a linked list. The height of the tree will be equal to the number of elements, and the time complexity for insertion and deletion will become linear.