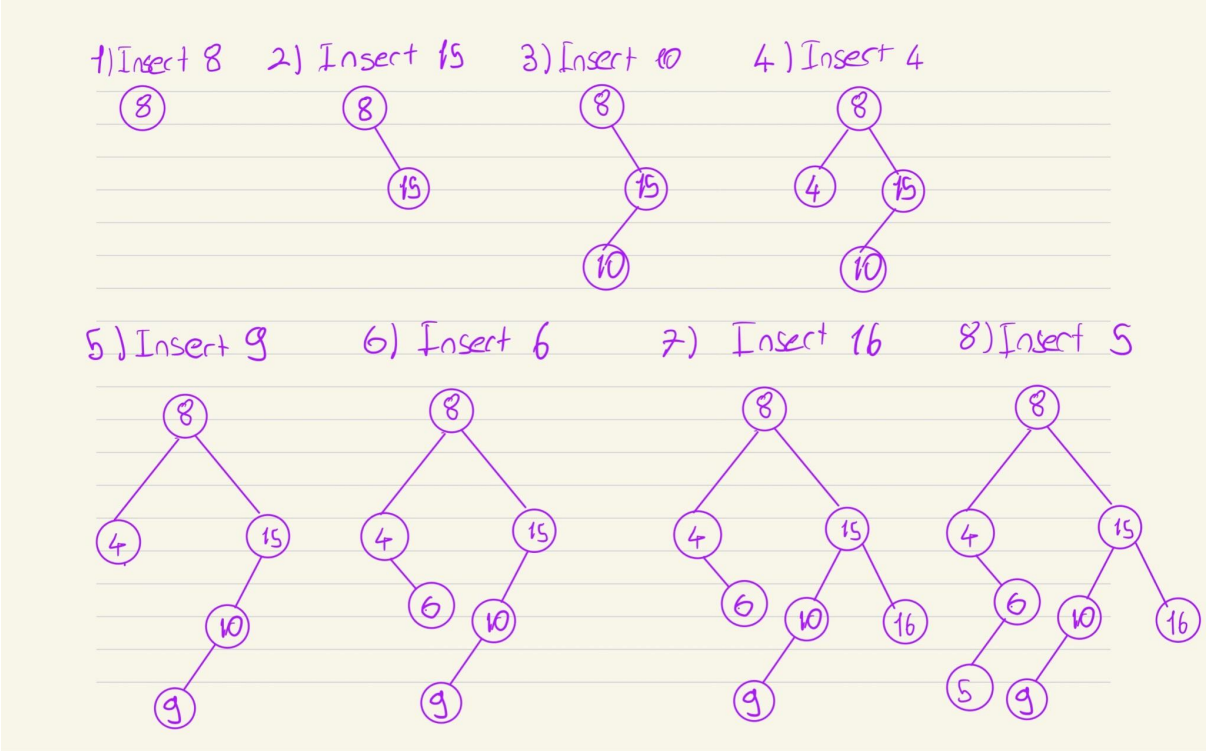
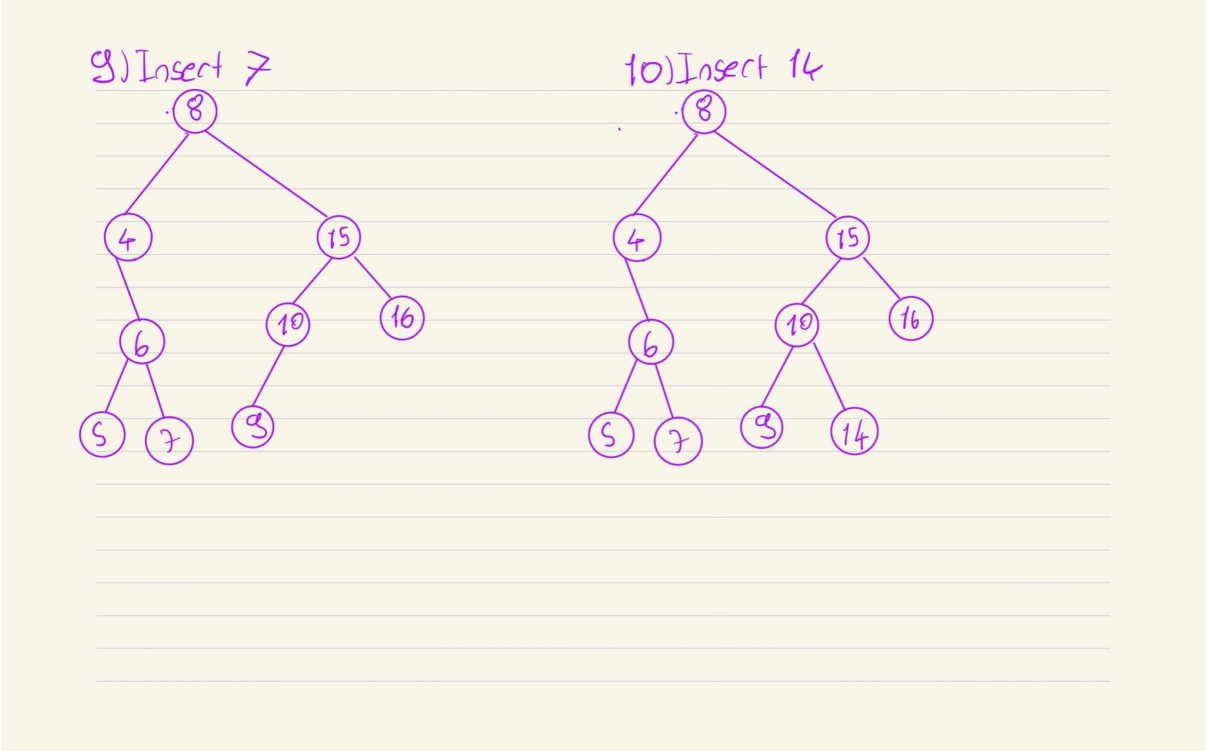
CS 202
Homework 2 - Binary Search Trees

Alper Bozkurt
21802766

Question-1

a) Insert 8, 15, 10, 4, 9, 6, 16, 5, 7, 14 into an empty binary search tree in the given order. Show the resulting BST after every insertion.





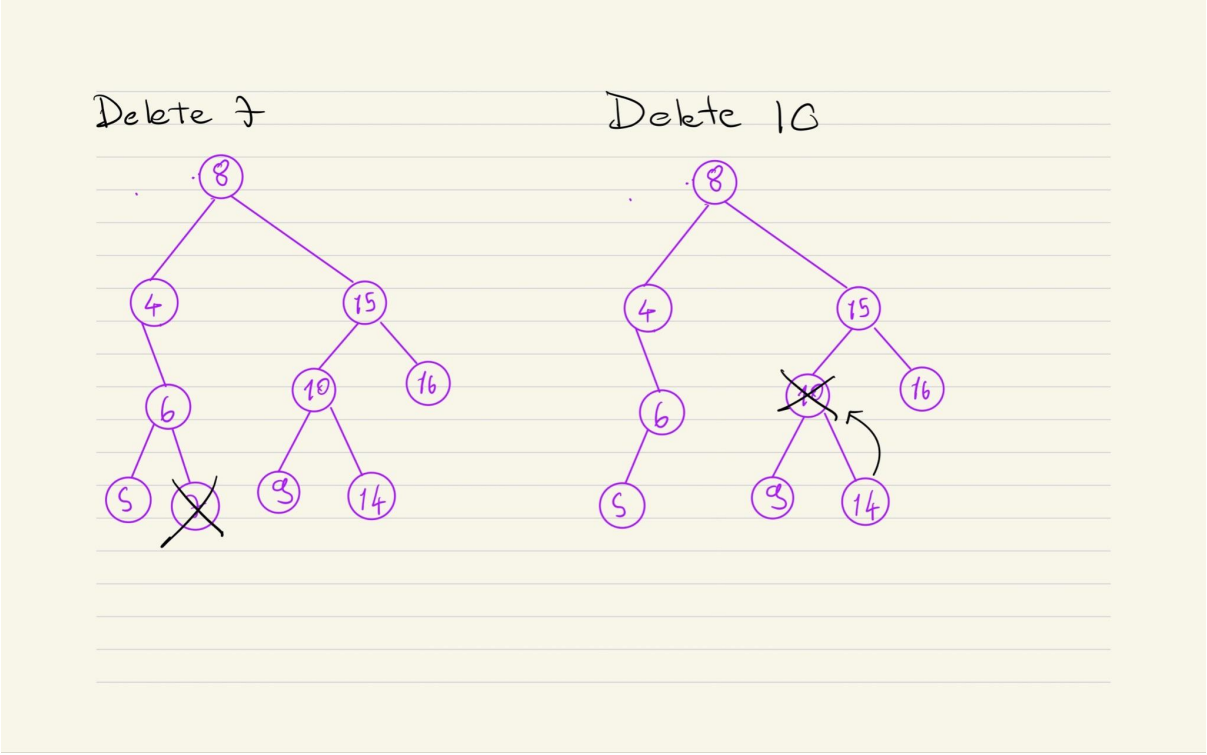
b) What are the preorder, inorder, and postorder traversals of the BST you have after (a)?

Preorder: 8 - 4 - 6 - 5 - 7 - 15 - 10 - 9 - 14 - 6

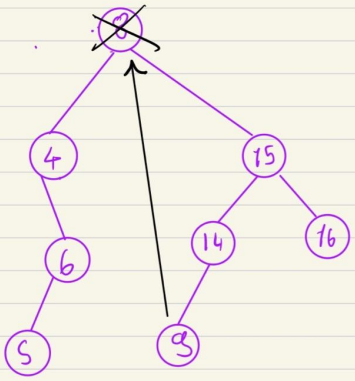
Inorder: 4 - 5 - 6 - 7 - 8 - 9 - 10 - 14 - 15 - 16

Postorder: 5 - 7 - 6 - 4 - 9 - 14 - 10 - 16 - 15 - 8

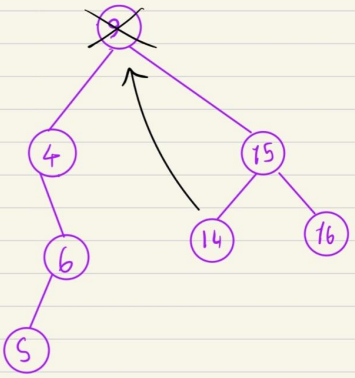
c) Delete 7, 10, 8, 9, 5 from the BST you have after (a) in the given order. Show the resulting BST after every deletion.



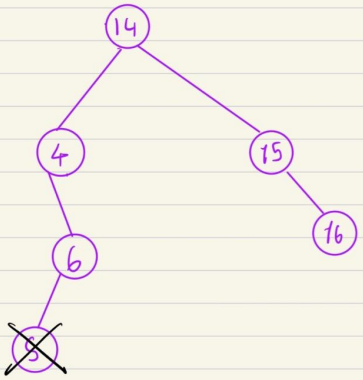
Delete 8

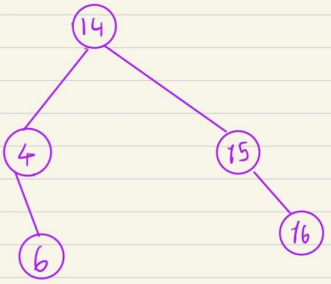


Delete 9



Delete 5





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

```
int findMin(BSTNode* treePtr)
    if(treePtr == NULL)
        return -1
    else if(treePtr -> left == NULL)
        return treePtr -> item
    else
        findMin(treePtr -> left)
```

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

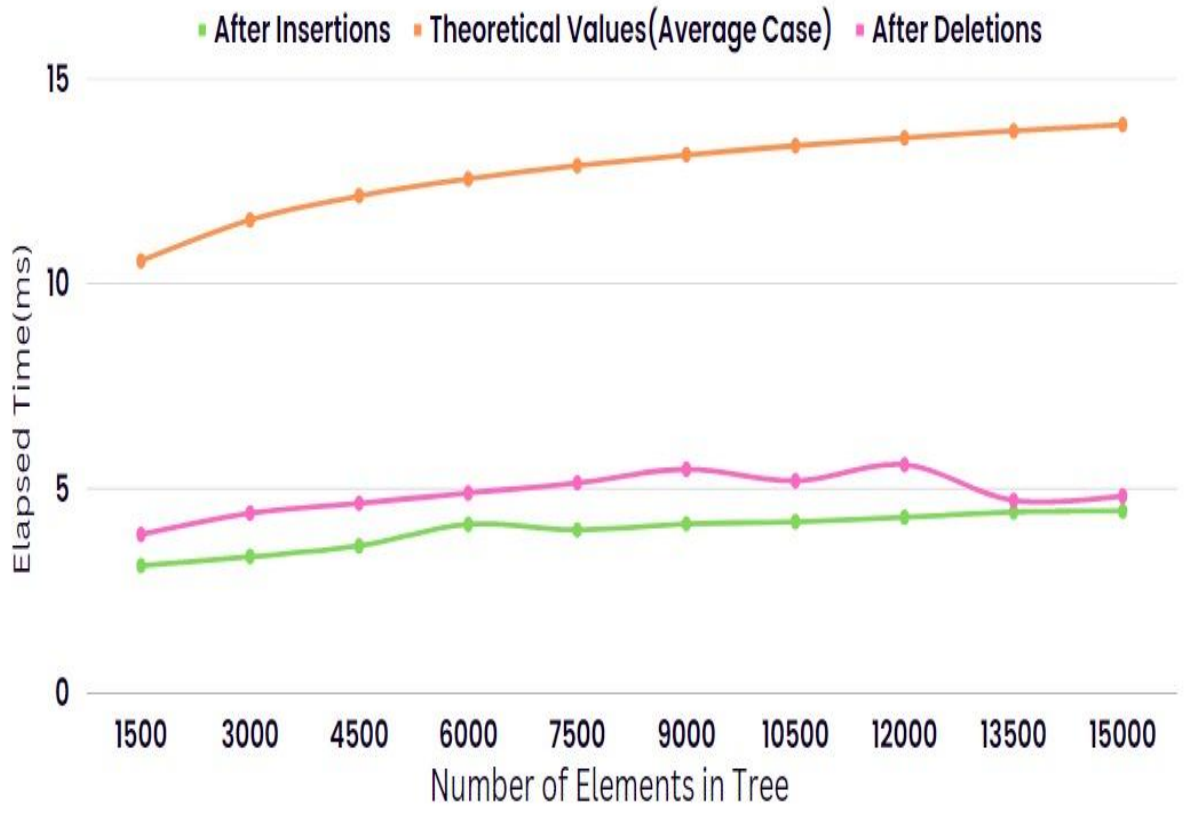
Maximum Height = n

Minimum Height = $\log_2(n+1)$

Question-3

Theoretical result of insertion and deletion is done according to the average case in that situation which is O(logn) when n numbers of elements in the tree. Both are recursive algorithms, so it is efficient. However, the empirical result becomes much lower compared to the theoretical one, even if it is manipulated by multiplying ten to interpret and show with ease. Random numbers between zero and the size of the array are added, so random numbers have limitations, and less diversity in numbers could be the explanation for that efficiency. Because the theoretical result is formed by the numbers that go to infinity; in other words, no limitation. The reason also could be the balance of that tree. With limited numbers, balance is more probably occurred compared to the random number. Since the balance tree's height is lower, the program uses less effort the insert or delete elements. The graph is only shown for the average case because the worst case, which is O(n), results are much higher. Therefore, comparison becomes meaningless. But for both cases graph will be incremented because a high number of values will take more time to insert or delete. That is the other reason for choosing the average case. In the worst case, incrementation is directly proportional.

If sorted numbers are inserted, the binary search tree becomes linear, in other words, unbalanced. Thus, in searching, because there is no branch that facilitates the search, it becomes the worst case. Adding the last element happens by visiting all the elements, so it is O(n) complexity.


After Insertions
Theoretical Values(Average Case)
After Deletions
15
10
5
0
Elapsed Time(ms)
1500
3000
4500
6000
7500
9000
10500
12000
13500
15000
Number of Elements in Tree