Title: Binary Search Tree
Author: Deniz Çalkan
ID: 21703994
Section: 1
Assignment: 2
Description: Answers of questions 1,2 and 3.

Q1a)

Preorder: <root><left><right>

* , -, 13, 1, 3, 2, +, 4, /, 21, 7

Inorder: <left><root><right>

13, -, 3, ^, 2, *, 4, +, 21, /, 7

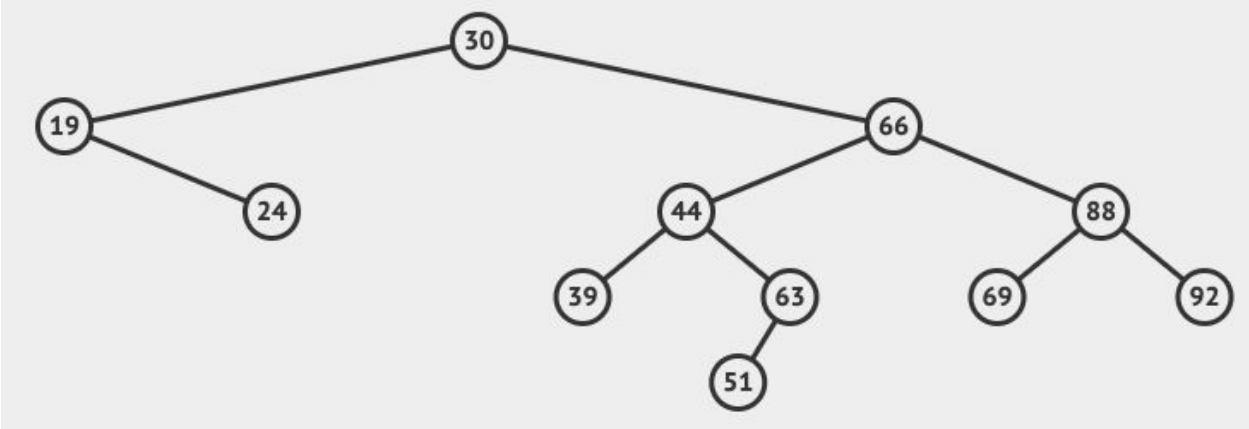
Postorder: <left><right><root>

13, 3, 2, 1, -, 4, 21, 7, /, +, *

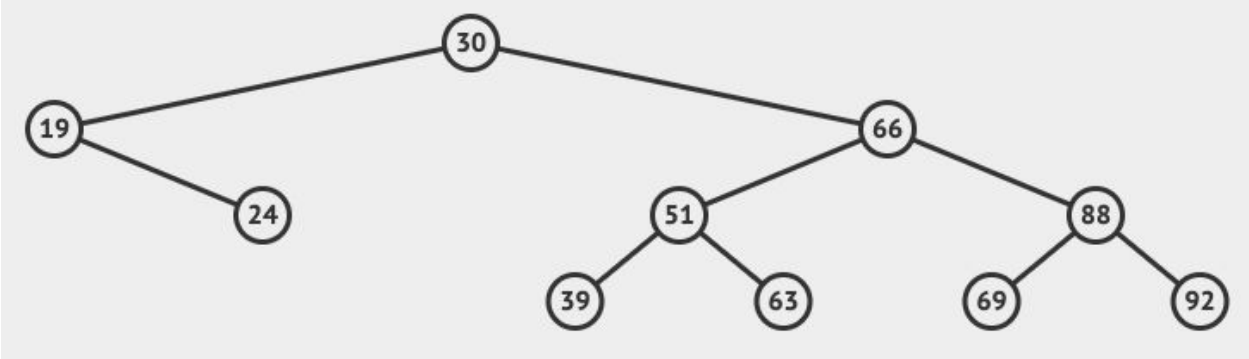
Computed solution using the inorder traversal: 28 (calculated each subtree first)

Q1b)

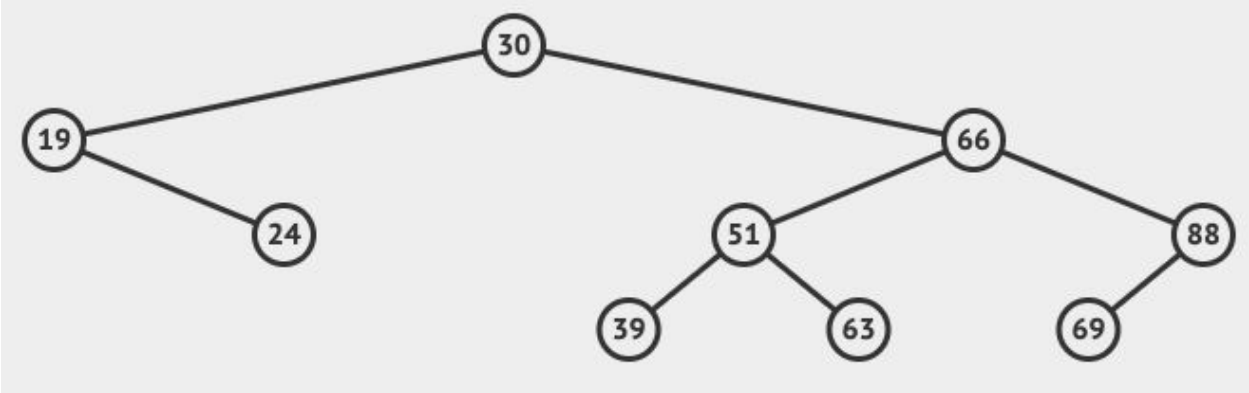
Final tree after all insertions:



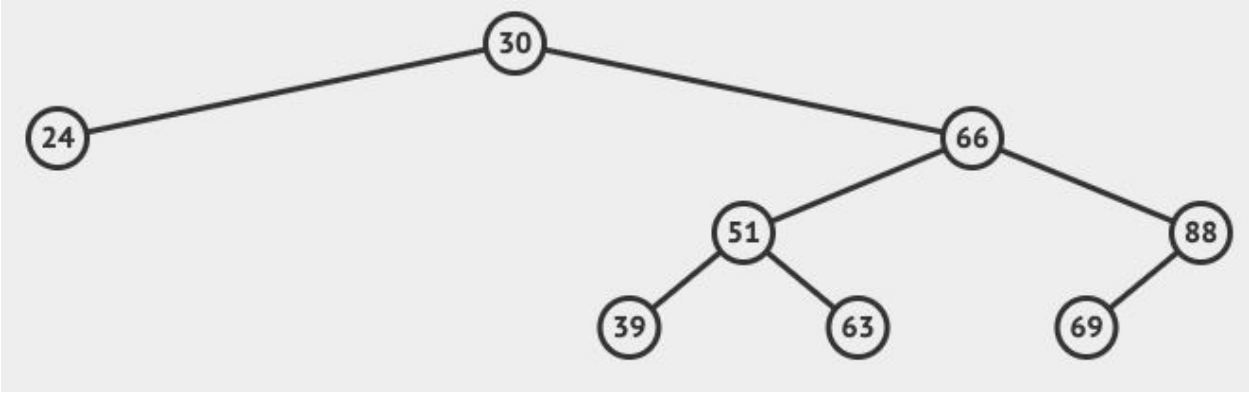
After deleting 44:



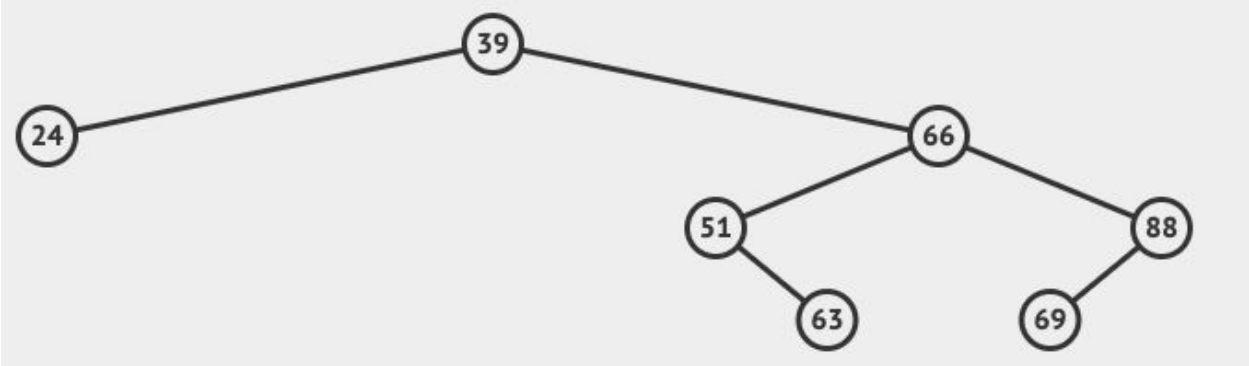
After deleting 92:



After deleting 19:



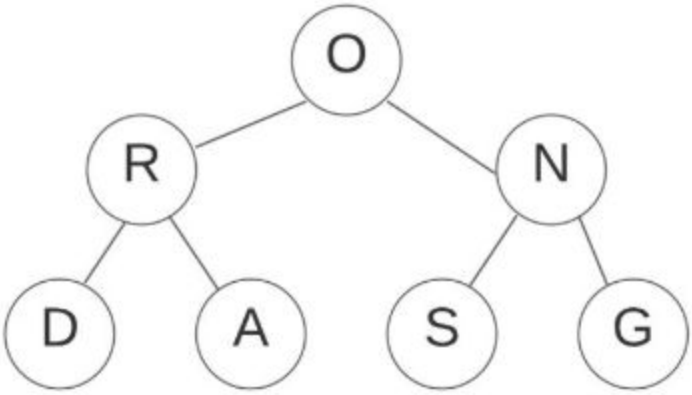
After deleting 30:



Q1c)

Inorder traversal: D, R, A, O, S, N, G

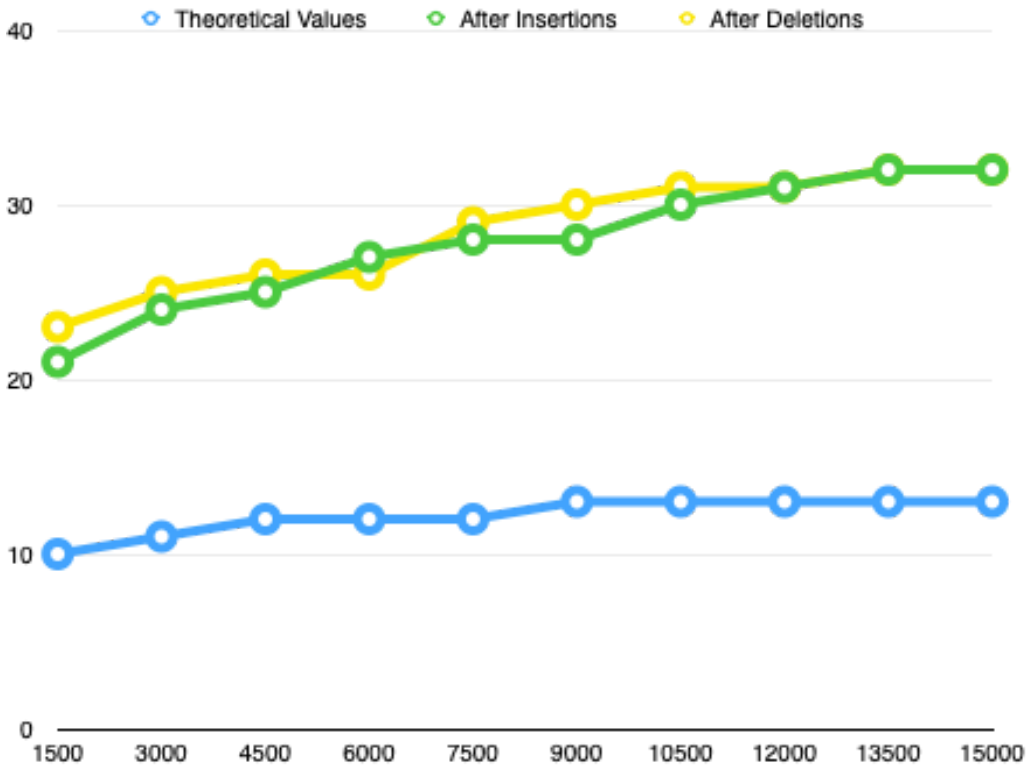
Full Binary Tree reconstructed from its traversal:



Q2) Output of the Program

```
Tree after insertions
42 19 11 8 6 13 22 35 32 28 56 45 94 81 63
Required nodes to be a full tree
48
Tree after deletions
42 22 11 8 6 13 35 32 28 63 45 94 81
Median of BST
35
Mirrored Tree
42 63 94 81 45 22 35 32 28 11 13 8 6
------------------------------------------------------------
Part e - Height analysis of Binary Search Tree - part 1
------------------------------------------------------------
Tree Size     Tree Height
------------------------------------------------------------
1500             21
3000             24
4500             25
6000             27
7500             28
9000             28
10500             30
12000             31
13500             32
15000             32
------------------------------------------------------------
Part e - Height analysis of Binary Search Tree - part 2
------------------------------------------------------------
Tree Size    Tree Height
------------------------------------------------------------
13500             32
12000             31
10500             31
9000             30
7500             29
6000             26
4500             26
3000             25
1500             23
0              0
Program ended with exit code: 0
```

Q3)



In the graph, theoretical values were obtained from the min height of a binary search tree with n nodes which is floor(logn). The reason for choosing min height instead of max height which is (n -1) is to be able to compare my results with the theoretical results easier. My results are not too far from the theoretical result but they have a larger height because I used random numbers instead of fixed numbers to fill the tree. Also my results are between two theoretical results which are min height and max height.

If we insert sorted numbers instead of random numbers to a tree, the height will be very close or the same with the theoretical result for max height which is (n-1). Since we will be inserting only in the same direction, if the array is sorted in ascending order then we will insert to the right all the time and if the array is sorted in descending order then we will insert to the left all the time. Which will give us the max height.