* Title : Binary Search Trees

* Author : Faaiz Ul Haque

* ID: 21503527

* Section : 2

* Assignment : 2

* Description : Questions 1 & 3

Question 1 – 15 points

A) [5 points] What are the preorder, inorder, and postorder traversals of the binary tree below:

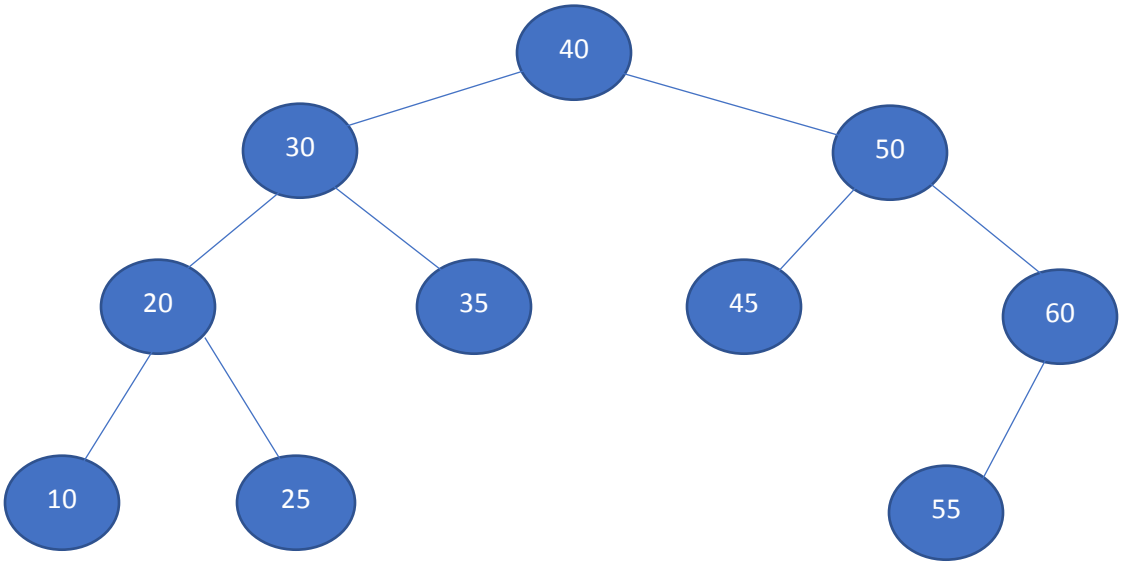
Preorder: RGLAOHITM

Inorder: ALGORITHM

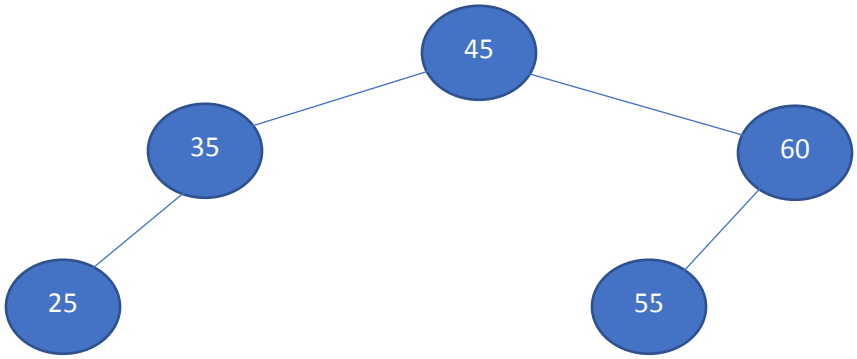
Postorder: ALOGTIMHR

B) [5 points] Insert 40, 50, 45, 30, 60, 55, 20, 35, 10, 25 to an empty Binary Search Tree. Then delete 10, 40, 50, 20, 30.

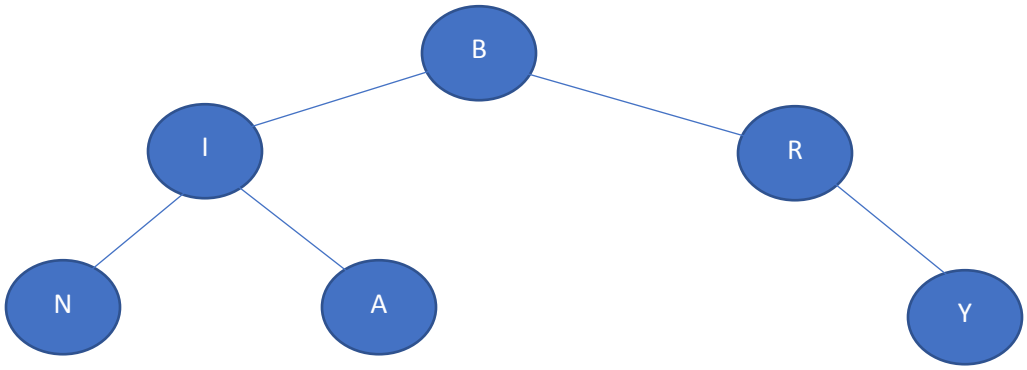
After all insertions: 40, 50, 45, 30, 60, 55, 20, 35, 10, 25



After all deletions: 10, 40, 50, 20, 30.



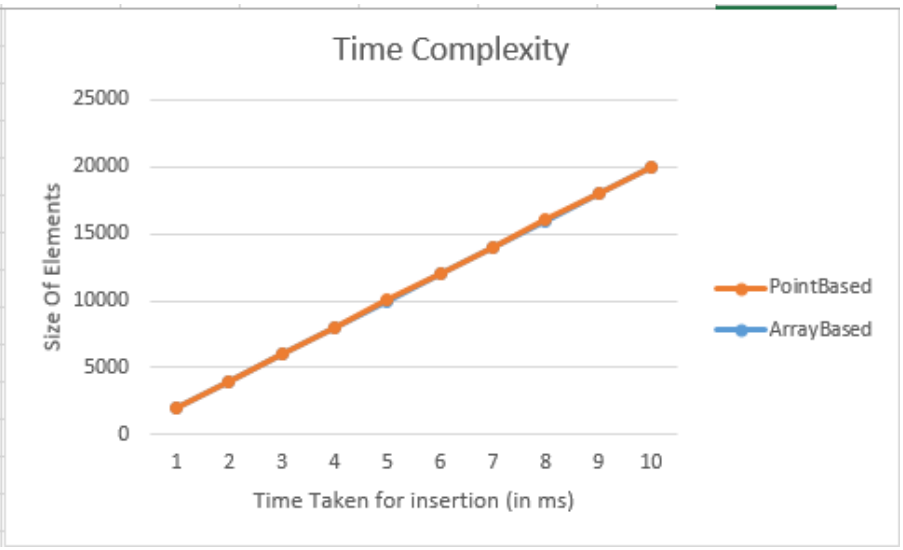
(c) [5 points] A binary tree has a pre-order traversal of B, I, N, A, R, Y and postorder traversal of N, A, I, Y, R, B. What is its inorder traversal? Reconstruct the tree from those traversals and draw it.



Inorder Traversal: NIABRY

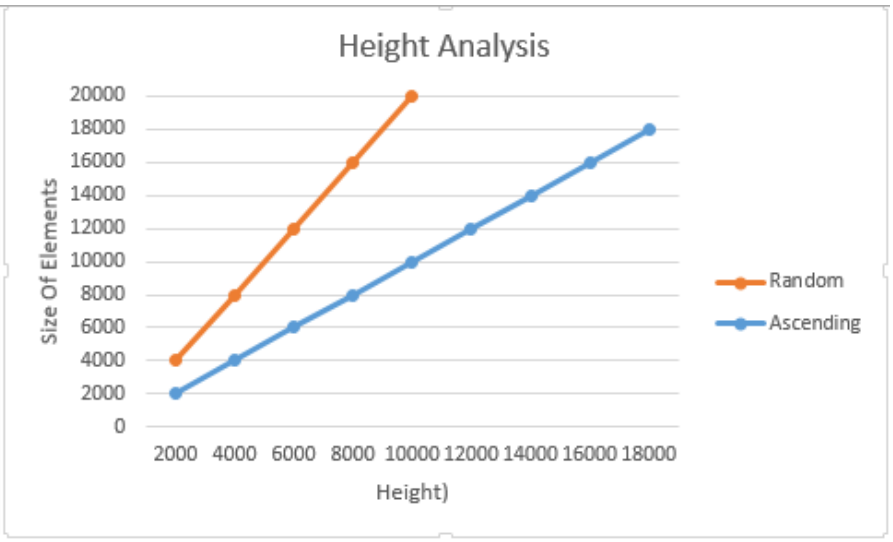
Question 3 – 15 points

2000	0	0
4000	0	0
6000	15	0
8000	0	0
10000	0	16
12000	0	0
14000	15	0
16000	0	22
18000	0	0
20000	16	16



Although invisible, the two graphs are on top of each other as they are hardly any different in insertion time complexity. To conclude my results are very accurate when compared to the real ones. Theoretical and practical results are almost the same. The difference in speed of the implementation is more or less the same when it comes to insertion however the Array is a little faster than pointers. Also the pointer based implementation takes up a lot of space. But array isn't a good choice for this because of the size problem where you have to keep changing it. So this reason makes the pointer based linked list a better choice for implementing binary search tree.

2000	1958	2000
4000	2337	4000
6000	3408	6000
8000	7917	8000
10000	6069	10000
12000	9783	12000
14000	9687	14000
16000	15890	16000
18000	17704	18000
20000	16225	20000



For this the average height for ascending random integers seemed to be nearly n/2 which we can break down to simply O(N). In the ascending case, it is the worst possible thing that could happen for Bst insertions and all nodes will be found on the right side leaving us with a linear data and a height as the same size as the whole array. In order to make sure the worst case doesn't happen we should try to keep balanced trees. So that insertion and other traversal processes don't take as long as O(N)