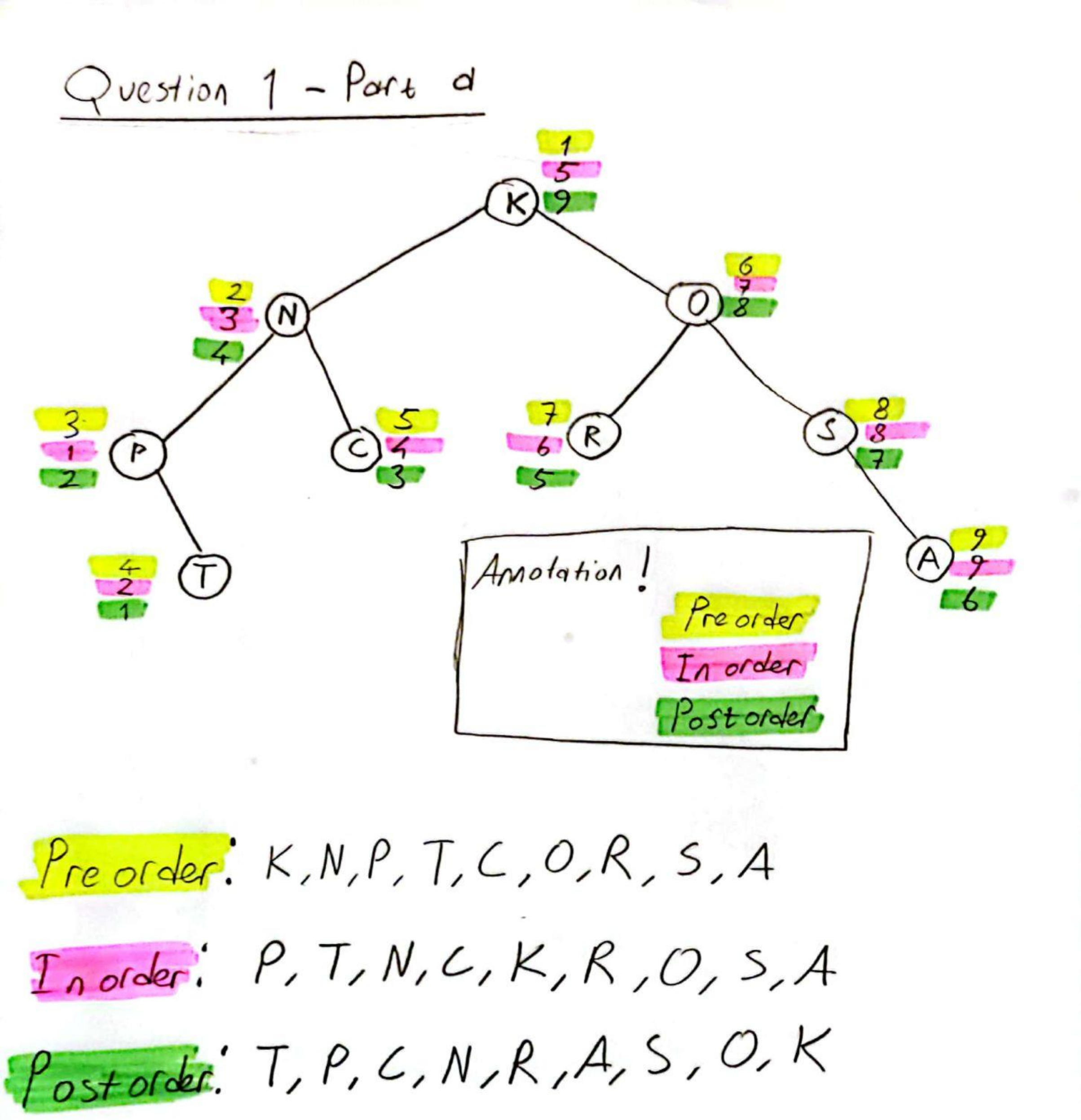
CS-202 HOMEWORK 2

Full Name: Arda İynem
ID: 22002717
Section: 1
Assignment: 2

Question 1 - Part a



Question 1 - Part a

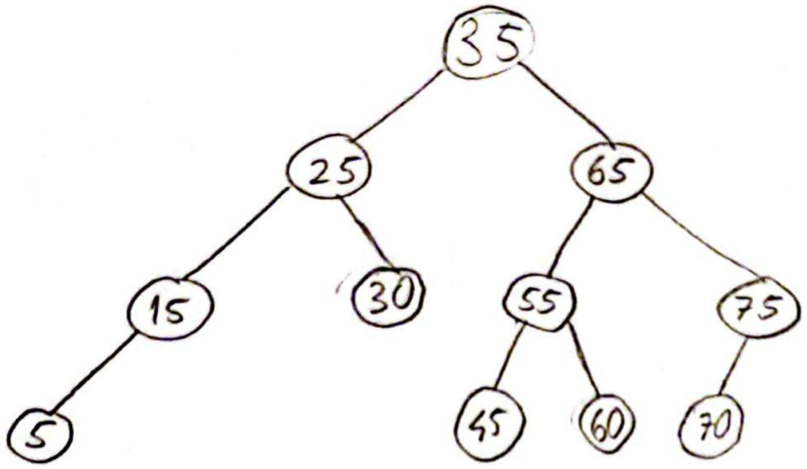
Pre order: K, N, P, T, C, O, R, S, A

In order: P, T, N, C, K, R, O, S, A

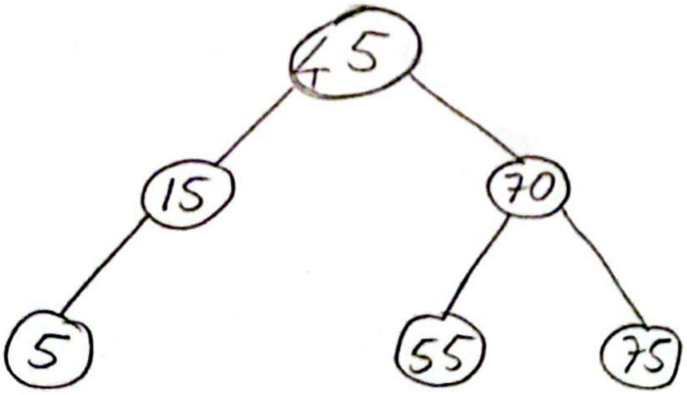
Post order: T, P, C, N, R, A, S, O, K

Question 1 - Part b

Question 1- Part b

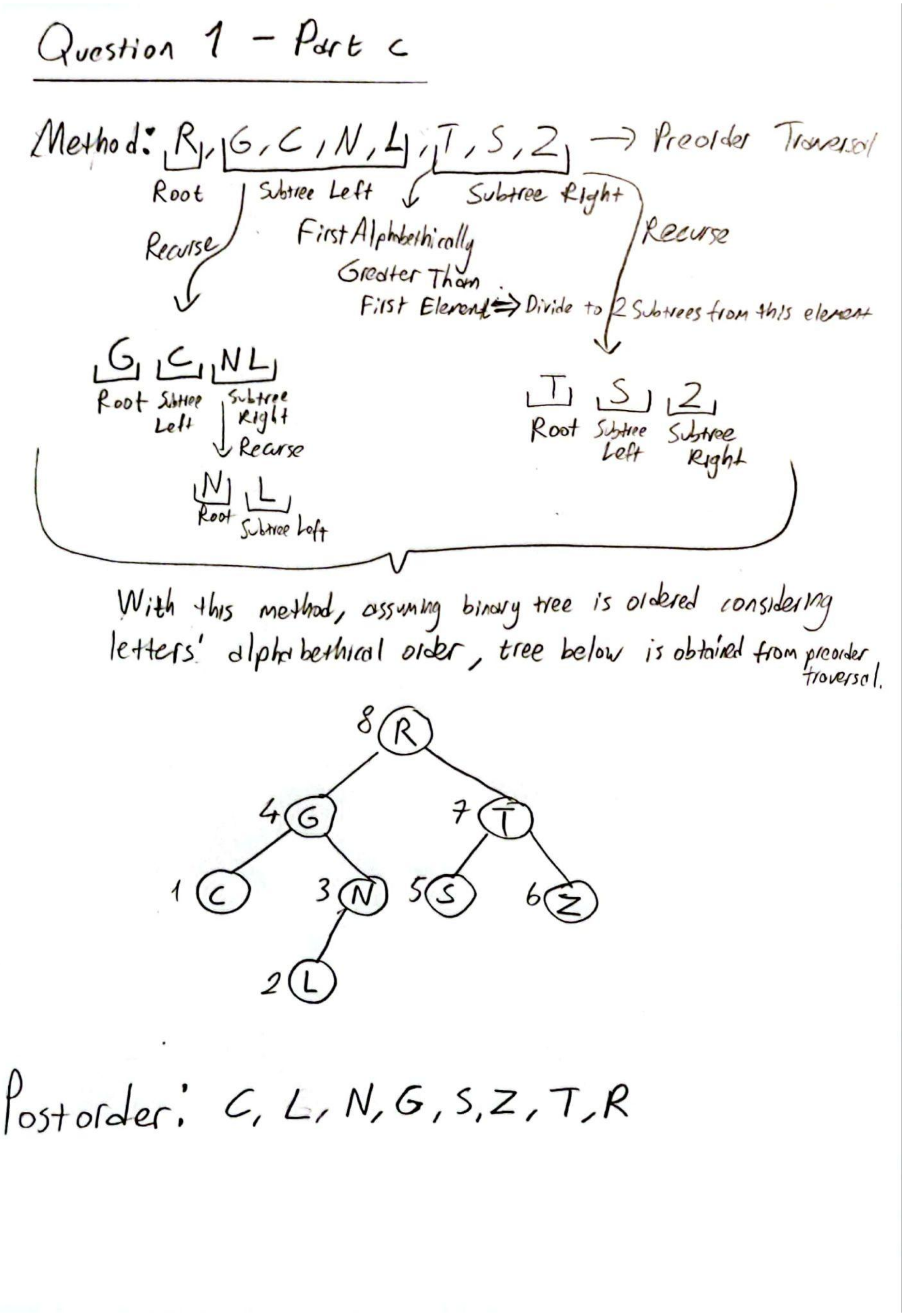


Final Tree After Insertions



Final Tree After Deletions

Question 1 - Part c



Question 1 - Part c

Method: $\underbrace{R}_{\text{Root}}, \underbrace{G, C, N, L}_{\text{Subtree Left}}, \underbrace{T, S, Z}_{\text{Subtree Right}}$ → Preorder Traversal

Recurse

First Alphabethically Greater Than First Element ⇒ Divide to 2 Subtrees from this element

Recurse

$\underbrace{G}_{\text{Root}}\ \underbrace{C}_{\text{Subtree Left}}\ \underbrace{NL}_{\text{Subtree Right}}$

Recurse

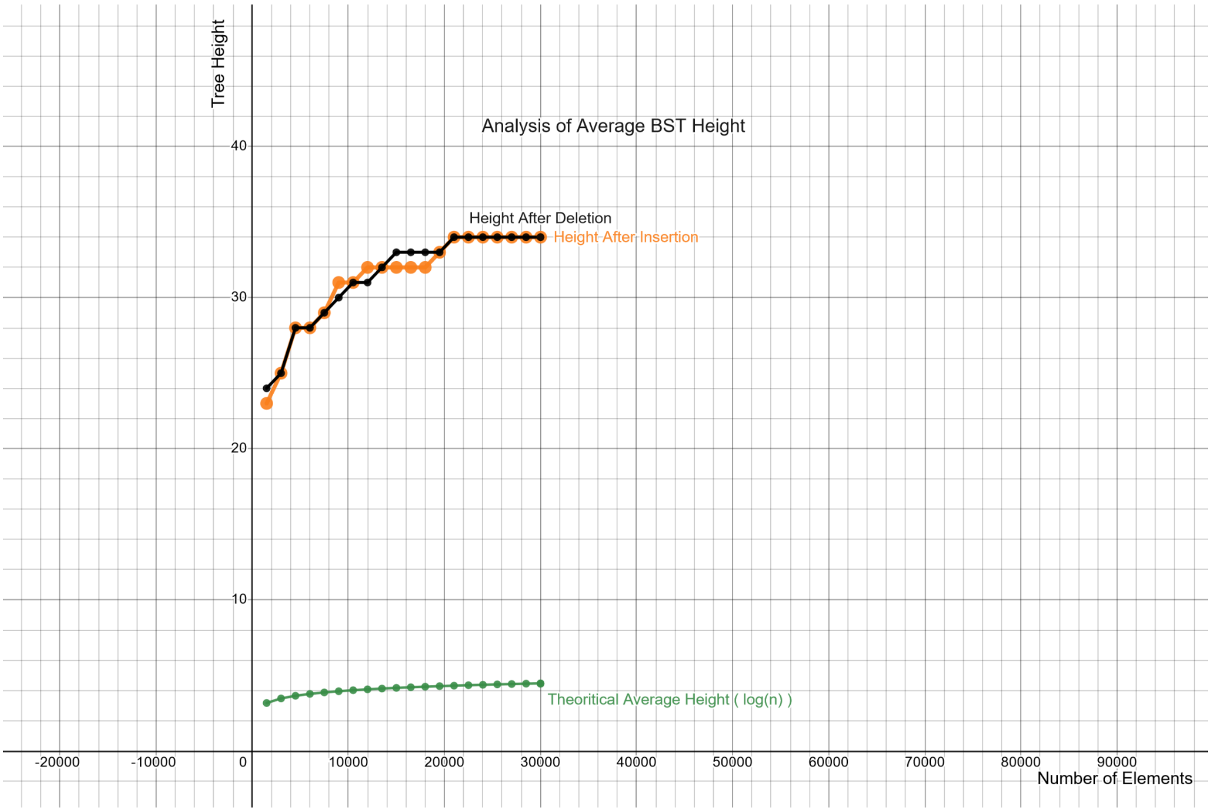
$\underbrace{N}_{\text{Root}}\ \underbrace{L}_{\text{Subtree Left}}$

$\underbrace{T}_{\text{Root}}\ \underbrace{S}_{\text{Subtree Left}}\ \underbrace{Z}_{\text{Subtree Right}}$

With this method, assuming binary tree is ordered considering letters' alphabethical order, tree below is obtained from preorder traversal.

Postorder: C, L, N, G, S, Z, T, R

Question 3



Obtained empirical height results are strongly related with the theoretical average BST height value log(n) since the obtained numerical data shows a logarithmic increase of height as the inserted total node number increased. Although, the empirical plot lines do not look like perfect theoretical log(n) line because of the small amount of data (whereas their line would look like that of log(n) much more with a bigger data set), the logarithmic increase rate of the empirical lines shows the resemblance between the empirical and theoretical results. Moreover the resemblance between random created array and shuffled deleted array pointed out the two average cases have very similar height results. Therefore my findings approved the values of the theoretical average height of BST.

If the inserted array was sorted, there would be 2 cases for ascending and descending sorted arrays:
Case 1: If the inserted array is sorted ascendingly, the first element of array would be the root and each node inserted after would be the right child node of the last inserted node since each new inserted node is greater than all of the current nodes in the tree. Therefore each insertion would increase the height by one and the final height would be equal to the maximum height of the array, namely the amount of the inserted nodes, which equals the inserted array’s size.

Case 2: If the inserted array is sorted descendingly, the same process in Case 1 would happen with a minor difference that each new inserted node would be the left child of the last inserted node with the same logic explained in Case 1. Even though the final tree would be symmetric to the Case 1, the height would be again equal to the size of the inserted array.