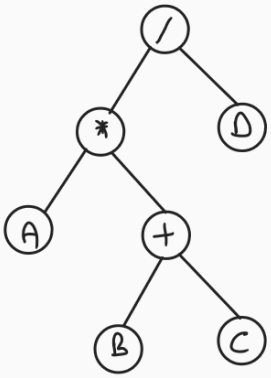
Bilkent University
CS202 - HW2
Section 3

Hacı Çakın
21802641

Question 1

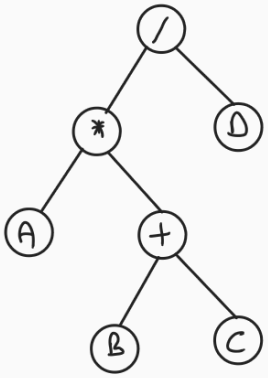
Part A

Prefix



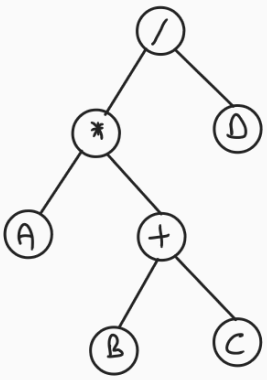
/*A+BCD

Infix



A*(B+C)/D

Postfix



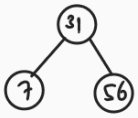
ABC+*D/

Part B

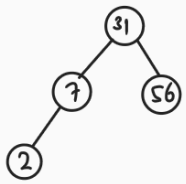
Insert 31

Insert 7

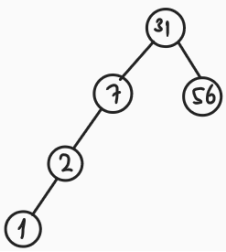
Insert 56



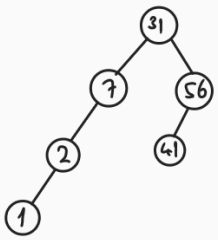
Insert 2



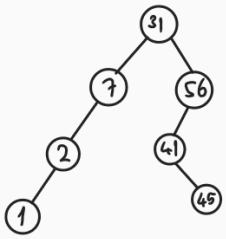
Insert 1



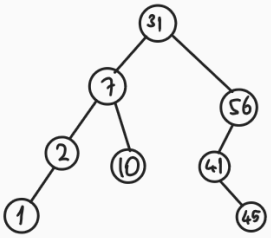
Insert 41



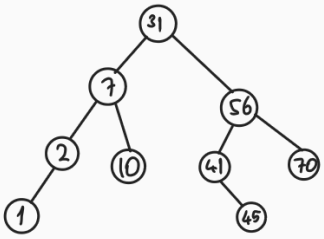
Insert 45



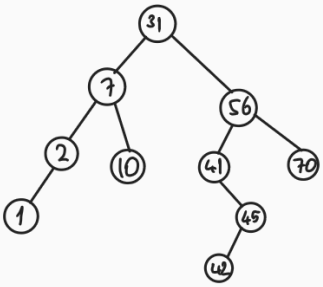
Insert 10



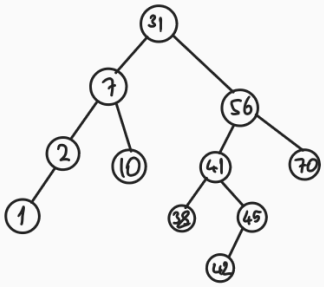
Insert 70



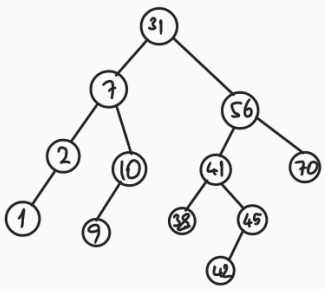
Insert 42



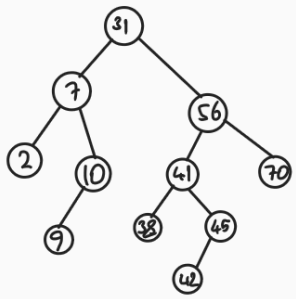
Insert 38



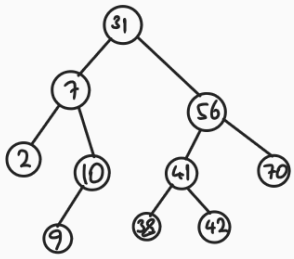
Insert 9



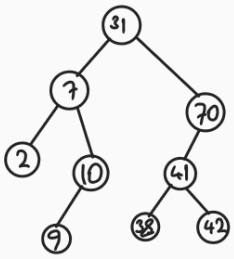
Delete 1



Delete 45

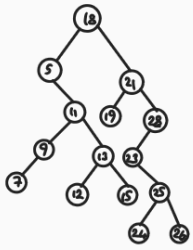


Delete 56



Part C

Preorder → 18, 5, 11, 9, 7, 13, 12, 15, 21, 19, 28, 23, 25, 24, 26



Postorder → 7, 9, 12, 15, 13, 11, 5, 19, 24, 26, 25, 23, 28, 21, 18

Question 3

A) Analysis Of levelorderTraverse method

Implementation of this contains recursive function. For each level, it calls overloaded one and it starts to work recursively. It travels all the nodes until its level is suitable. Worst case is the for each node recursive function might travel all nodes. Therefore, for the worst case big-O is O(n^2). Better algorithm can be written with using queue. While visiting a node, its child can be put in queue. In queue, since FIFI is working, the order will not be broken. Therefore, it is not visiting more than one for each node, it is efficiency is O(n).

B) Analysis Of span method

Implementation of this span method also contains recursive function. In the span function which I overloaded to use recursively, function starts to travel BST from the root. According to valid condition, it cals itself. These conditions are explained in detail.

1) Node value can be equal to min value. In this situation, nodes in right child of this node can be in this range. However, nodes in the left child can not, therefore there is no requirement to pass there.
2) Node value can be equal to max value. In this situation, reverse of previous one is valid.
3) Node value can between the max and min. In this situation, both left and right child are the candidate therefore it call both of them.
4) Node value can be bigger than max value. In this situation, nodes in left child of the current node can be in this range. However, nodes in right child of current node can not be in this range so there is no requirement to call them.
5) Node value can be smaller than min value. In this situation, reverse of the 4th is valid.
6) Node is null, then return 0 and not call again.

For this implementation, function travels exactly same with the number of values in range a-b. Therefore, it is not travels whole the list unless whole list in the range. Worst case which is the whole list in the range has big-O O(n). In trees, we do not use any special space, therefore in order to understand how many value in the range, we have to

visit at least these values. There is another worst case situation even none of the values in range, It is if list has nodes same number with heigh of three, then whether these values are in the range or not, it is again takes 0(n). In the description of homework 2, there is a statement that your function should not visits all nodes. This function is not visiting all nodes except worst cases. It follows the logic which is smaller ones are in left subtree and larger ones are in right subtree.

G) Analysis Of mirror method

Implementation of the mirror method contains recursive parts as previous ones have. In the function, It calls itself for both right and left child. After calling, it replaces them. The logic is here is that, because root already called the mirror function for its right and left child, in its left and right subtrees, they also do same thing. Therefore, function start to replace from the leaves. After replacing is done, replacing continues for parent of children. The order of call causes this. In worst case, best and average cases, big-O is equal to O(n) because functions visits all nodes just one times. I don't think that there is no possibility to better big-o because in order to take mirror, we have to replace values which means we have to reach these values. Therefore, we have to reach n values which means its time complexity will be O(n).