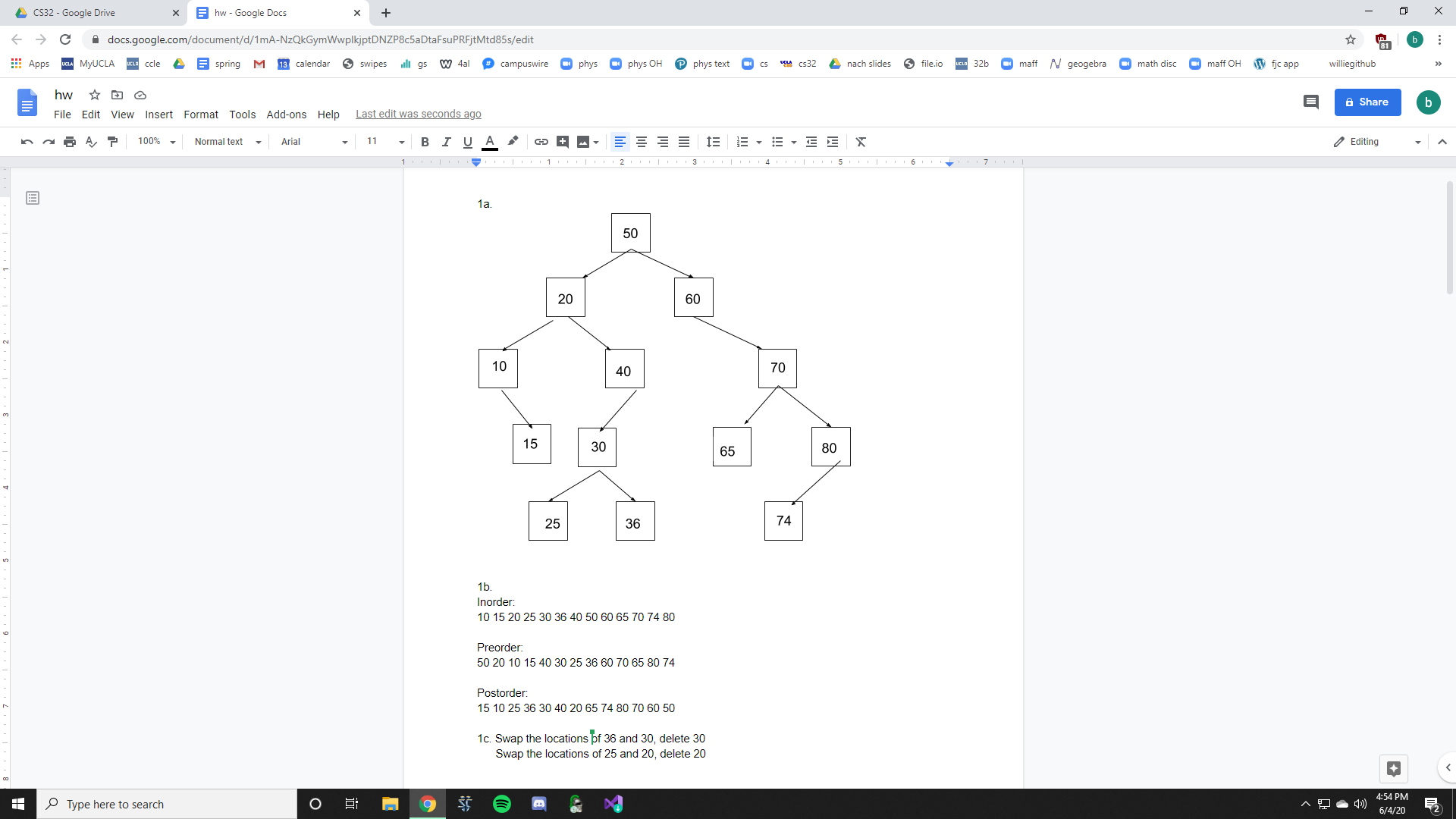
1a.



1b.

Inorder:

10 15 20 25 30 36 40 50 60 65 70 74 80

Preorder:

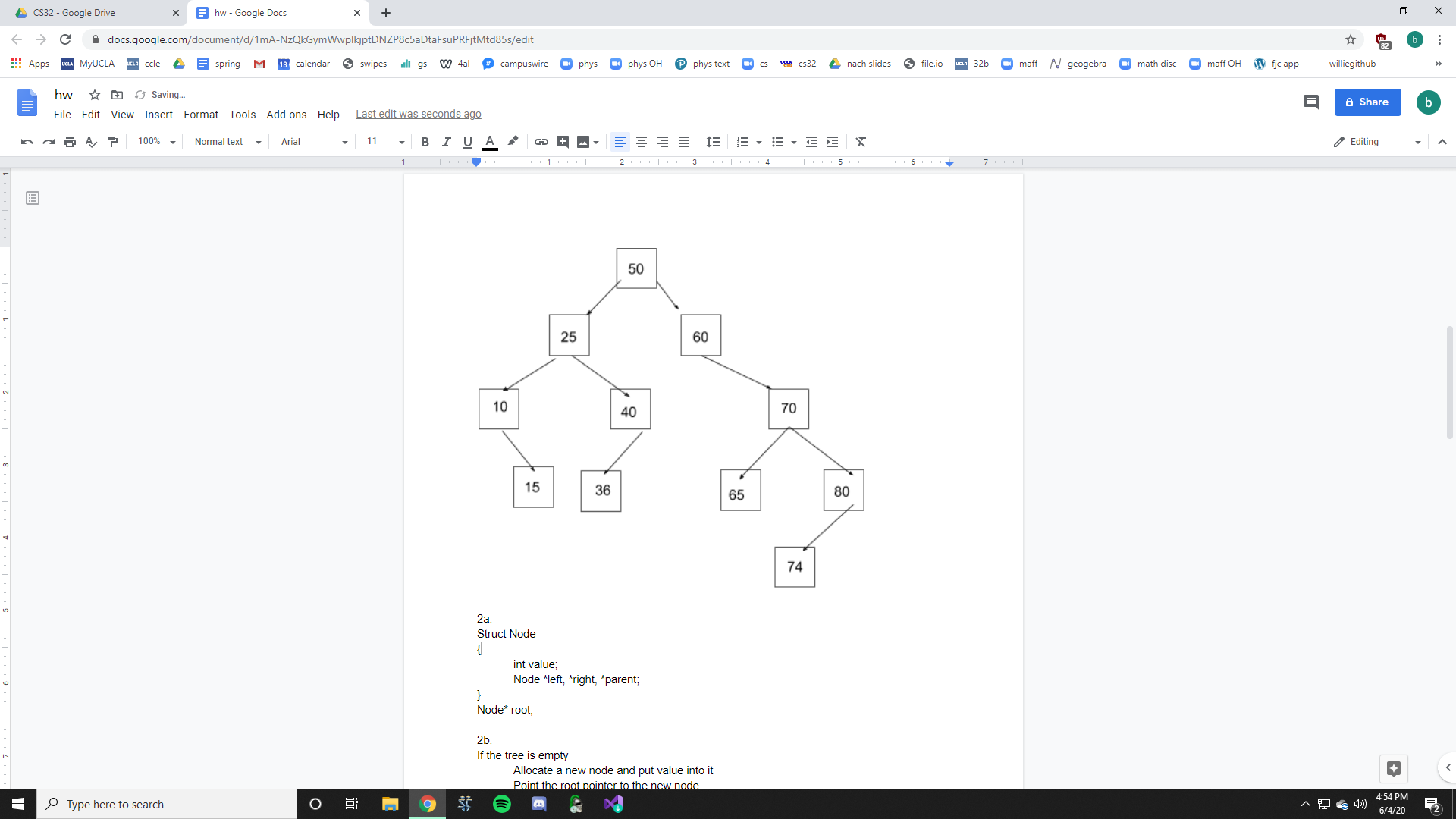
50 20 10 15 40 30 25 36 60 70 65 80 74

Postorder:

15 10 25 36 30 40 20 65 74 80 70 60 50

1c. Swap the locations of 36 and 30, delete 30

Swap the locations of 25 and 20, delete 20



2a.

Struct Node

{

int value;

Node \*left, \*right, \*parent;

}

Node\* root;

2b.

If the tree is empty

Allocate a new node and put value into it

Point the root pointer to the new node

Set parent ptr to nullptr

return

Set current node to root (start at root of tree)

While we havent reached a return statement (just make it infinite loop, the return statement will end the loop)

If value is equal to current node’s value

return

If value is less than current node’s value

If current node has a left node

Set current node to current’s left node (go left)

Else

Allocate a new node and put value into it

Set current node’s left pointer to new node

Set new node’s parent pointer to current node

return

Else if value is greater than current node’s value

If current node has a right node

Set current node to current’s right node (go right)

Else

Allocate a new node and put value into it

Set current node’s right pointer to new node

Set new node’s parent pointer to current node

return

3a.



3b. [7,5,6,3,0,1]

3c. [6,5,1,3,0]

4a. O(C+ S)

4b. O(log C + S)

4c. O(log C + log S)

4d. O(log S)

4e. O(1)

4f. O(log C + S)

4g. O(S log S)

4h. O(C log S)