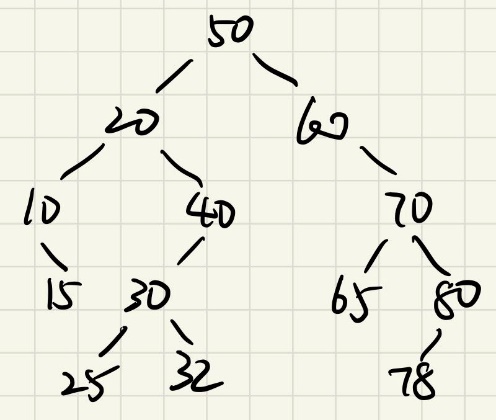
1. a.



b.

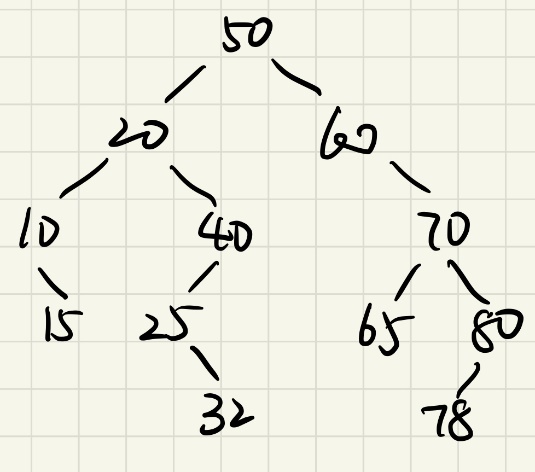
In-order: 10 15 20 25 30 32 40 50 60 65 70 78 80

Pre-order: 50 20 10 15 40 30 25 32 60 70 65 80 78

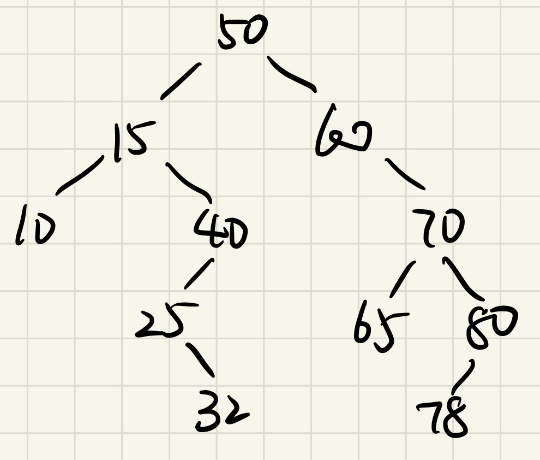
Post-order: 15 10 20 25 32 30 40 20 65 78 80 70 60 50

c.

After delete node 30:



Then after delete node 20:



2. a.

struct node {

int val;

node\* leftPtr;

node\* rightPtr;

node\* parentPtr;

}

b.

Input: an integer value to be inserted

Output: none

if root does not exist:

insert a new node with the input value as root

set parent, left, and right pointer of the new node to null

return

set current traversed node as root

while true:

if value of the current node equals to the input value:

return

if value of the current node is smaller than the input value:

if the current node has a left child:

set the current node as its left child

else:

insert a new node with the input value to the left child of the current node

set the left pointer of the current node as the new node

set the parent pointer of the new node as the current node

return

else if value of the current node is larger than the input value:

if the current node has a right child:

set the current node as its right child

else:

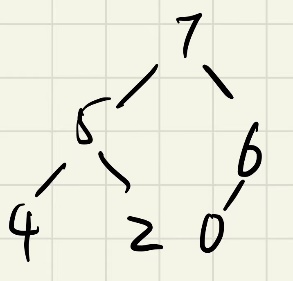
insert a new node with the input value to the right child of the current node

set the right pointer of the current node as the new node

set the parent pointer of the new node as the current node

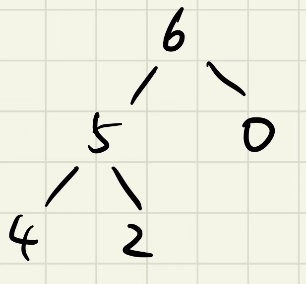
return

3. a.



b. [7, 5, 6, 4, 2, 0]

c.



4.

a. O(C + S)

b. O(log C + S)

c. O(log C + log S)

d. O(log S)

e. O(1)

f. O(log C + S)

g. O(S log S)

h. O(C log S)