Cody Hubbard HW 5 004 843 389

1.(a) insert 80, 65, 75, 15, 35 and 25



1.(b)

Pre Order - 50, 20, 10, 15, 40, 30, 25, 35, 60, 70, 65, 80, 75

In Order - 10, 15, 20, 25, 30, 35, 40, 50, 60, 65, 70, 75, 80

Post Order - 15, 10, 25, 35, 30, 40, 20, 65, 75, 80, 70, 60, 50

1.(c) delete 30 and then 20



2. (a)

struct node

{

node \*parentPtr, \*leftchild, \*rightchild;

int m\_data;

};

2. (b)

To Insert a newnode by passing in the newnode and a startpointer

If the startpointer is null set the startpointer to the newnode and set the new node’s parent to startpointer, then return.

Else if the startpointer is not null

If the newnode has value greater than startpointer’s node

Call the same insert function, passing in the newnode and startpointer’s right-childpointer as the startpointer

If the newnode has value lesser than or equal to startpointer’s node

Call the same insert function, passing in the newnode and startpointer’s left-childpointer as the startpointer

3. (a)



3.(b)

8 | 3 | 6 | 0 | 2 | 4

3.(c)

6 | 3 | 4 | 0 | 2

4.

1. O(C+S)
2. O(logC + S)
3. O(logC + logS)
4. O(C + logS)
5. O(C + S)
6. O(logC + S)
7. O(C + SlogS + S) //find correct c, use mergesort, display all s
8. O(C\*logS)