1a. 50

20 60

10 40 70

15 30 65 80

25 35 75

1b. Pre: 50 20 10 15 40 30 25 35 60 70 65 80 75

Post: 15 10 25 35 30 40 20 65 75 80 70 60 50

In: 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80

1c. 50

15 60

10 40 70

25 65 80

35 75

2a.

struct Node

{ int value;

Node\* right, left, parent;

};

2b.

If the tree is empty

Allocate a new node and put V into it

Point the root pointer to our new node. DONE!

Start at the root of the tree

While we’re not done…

If V is equal to current node’s value, DONE! (nothing to do...)

If V is less than current node’s value

If there is a left child, then go left

ELSE allocate a new node and put V into it, and

set current node’s left pointer to new node, and

set new node’s parent pointer to current. DONE!

If V is greater than current node’s value

If there is a right child, then go right

ELSE allocate a new node and put V into it,

set current node’s right pointer to new node, and

set new node’s parent pointer to current. DONE!

3a.

9

3 8

0 2 4 6

3b. |9|3|8|0|2|4|6|

3c.

8

3 6

0 2 4

4. a. O(C + S) b. O(logC + S) c. O(logC + logS) d. O(1 + logS) e. O(1 + 1)

f. O(logC + S) g. O(1 + SlogS) h. O(C + logS)