1a.

50

20 60

10 40 70

15 30 65 80

25 34 76

1b.

in-order: 10, 15, 20, 25, 30, 34, 40, 50, 60, 65, 70, 76, 80

pre-order: 50, 20, 10, 15, 40, 30, 25, 34, 60, 70, 65, 80, 76

post-order: 15, 10, 25, 34, 30, 40, 20, 65, 76, 80, 70, 60, 50

1c.

delete 30

50

20 60

10 40 70

15 25 65 80

34 76

delete 20

50

15 60

10 40 70

25 65 80

34 76

2a.

struct Node {

int value;

Node\* left, right, parent;

};

2b.

void insert(int value) {

if the tree is empty

create node with value and all Node pointers = nullptr

set root = the node that was just created

return

create a pointer to root

while true (will return when Node is inserted)

if the value trying to be inserted == value of the Node pointed to

return // value already there

if value of the Node pointed to is less than the value trying to be inserted

move pointer to point to the left child

else

create a new node at the left child

return

else

if the pointer’s right child is not nullptr

move pointer to point to the right child

else

create a new node at the right child

return

}

3a.

7

3 5

0 2 4

3b.

|  |  |
| --- | --- |
| index | value |
| 0 | 7 |
| 1 | 3 |
| 2 | 5 |
| 3 | 0 |
| 4 | 2 |
| 5 | 4 |

3c.

5

3 4

0 2

4a. O(C+S)

4b. O(logC+S)

4c. O(logC+logS)

4d. O(logS)

4e. O(1)

4f. O(S+logC)

4g. O(SlogS)

4h. O(ClogS)