1a.

50

↙ ↘

20 60

↙ ↘ ↘

10 40 70

↘ ↙ ↙ ↘

15 30 64 80

↙ ↘ ↙

28 37 75

1b.

Inorder: 10, 15, 20, 28, 30,37, 40, 50, 60, 64, 70, 75, 80

Preorder: 50, 20, 10, 15, 40, 30, 28, 37, 60, 70, 64, 80, 75

Postorder: 15, 10, 28, 37, 30, 40, 20, 64, 75, 80, 70, 60, 50

1c.

50

↙ ↘

15 60

↙ ↘ ↘

10 40 70

↙ ↙ ↘

37 64 80

↙ ↙

28 75

2a.

class Node

{

Node\* left;

Node\* right;

Node\* parent;

int data;

Node(int value): left(nullptr), right(nullptr), parent(nullptr), data(value){}

};

2b.

function insert(root, value)

if root is null

// Tree is empty, create the root node

root = new TreeNode(value)

return root

current = root

parent = null

while current is not null

parent = current

if value < current.data

current = current.left

else if value > current.data

current = current.right

else

// Value already exists in the tree, do nothing

return root

// Create the new node

newNode = new TreeNode(value)

// Set the parent pointer of the new node

newNode.parent = parent

// Attach the new node to the appropriate parent's child pointer

if value < parent.data

parent.left = newNode

else

parent.right = newNode

return root

3a.

7

5 6

2 1 3

3b.

0 1 2 3 4 5

7 5 6 2 1 3

3c.

0 1 2 3 4

6 5 3 2 1

4a. O(C + S)

4b. O(logC + S)

4c. O(logC + log S)

4d. O(log S)

4e. O(1)

4f. O(logC + S)

4g. O(S \* log S)

4h. O(C \* log S)