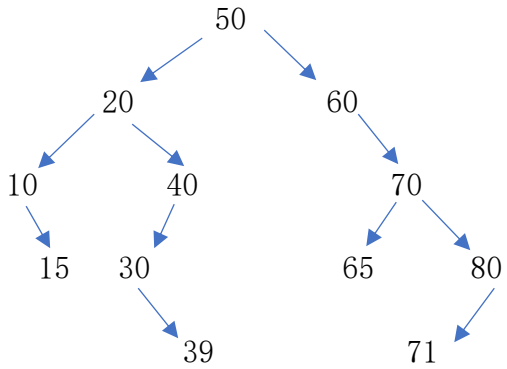
1

a)



b)

Inorder:

10 15 20 30 39 40 50 60 65 70 71 80

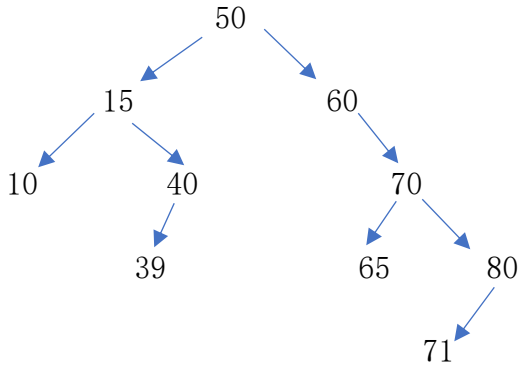
Preorder:

50 20 10 15 40 30 39 60 70 65 80 71

Postorder:

15 10 39 30 40 20 65 71 80 70 60 50

c)



2.

a)

```
struct Node
{
int value;
Node* parent;
Node* left;
Node* right;
}
```

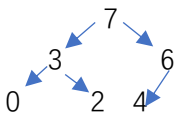
b)

```
Node* insertInorder(Node* subTreePtr, Node* newNodePtr)
if (subTreePtr is nullptr)
return newNodePtr
else if (subTreePtr->getItem() > newNodePtr->getItem())
{
tempPtr = insertInorder(subTreePtr->getLeftChildPtr(), newNodePtr)
subTreePtr->setLeftChildPtr(tempPtr)
tempPtr->setParent(subTreePtr)
}
else
{
tempPtr = insertInorder(subTreePtr->getRightChildPtr(), newNodePtr)
subTreePtr->setRightChildPtr(tempPtr)
tempPtr->setParent(subTreePtr)

}
return subTreePtr
```

3

a)



b) 7 3 6 0 2 4

index: 0 1 2 3 4 5

c) 6 3 4 0 2

4

a) O(C+S)
b) O(logC + S)
c) O(logC + logS)
d) O(1+logS)
e) O(1)
f) O(logC+S)
g) O(1+SlogS+S)
h) O(ClogS)