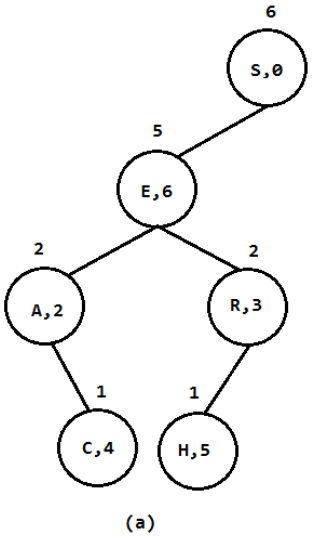
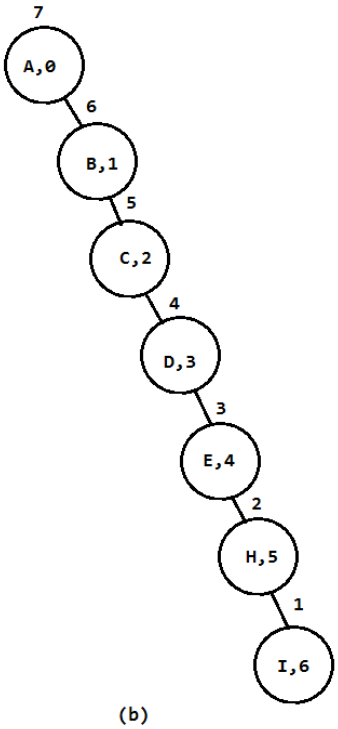
1. For each column provide the BST after doing the following operations, assume BST is initially empty.

a	b	c
put(“S”, 0) put(“E”, 1) put(“A”, 2) put(“R”, 3) put(“C”, 4) put(“H”, 5) put(“E”, 6)	put(“A”, 0) put(“B”, 1) put(“C”, 2) put(“D”, 3) put(“E”, 4) put(“H”, 5) put(“I”, 6)	put(“D”, 3) put(“B”, 1) put(“H”, 5) put(“A”, 0) put(“C”, 2) put(“E”, 4) put(“I”, 6)

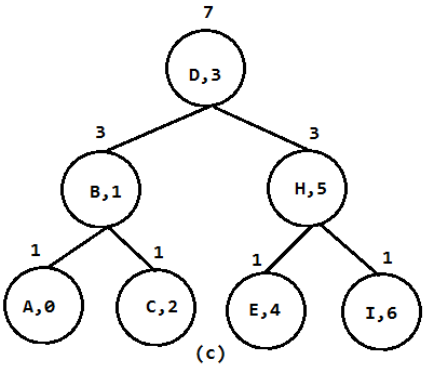
Answer:



(a)



(b)



(c)

2. Write a function to return the value associated with the largest key in a BST

Answer:

```
public Value max() {
        if (root == null) {
                return null;
        } else {
                return max(root).val;
        }
}

private Node max(Node node) {
        if (node.right == null) {
                return node;
        } else {
                return max(node.right);
        }
}
```

3. Write a function to return the height of a binary tree

- Height of a tree with single node is 0
- Height of a tree is the maximum height of the left and right sub-tree plus 1

Answer:

```
public int getHeight() {
        return getHeight(root);
}

private int getHeight(Node node) {
        if (node == null) {
                return -1;
        } else {
                int leftHeight = getHeight(node.left);
                int rightHeight = getHeight(node.right);
                int maxHeight = (leftHeight >= rightHeight ? leftHeight : rightHeight);
                return maxHeight + 1;
        }
}
```

5. Implement deleteMax(), which deletes the node with maximum key

Answer:

```
public void deleteMax() {
	if (root!=null) {
		root = deleteMax(root);
	}
}

private Node deleteMax(Node node) {
	if (node.right == null)  return node.left;
	node.right = deleteMax(node.right);
	node.n = size(node.left) + size(node.right) + 1;
	return node;
}
```

6. Implement the following methods:

private void printPreorder(Node node); // visit node, left tree, right tree

private void printPostorder(Node node); // visit left tree, right tree, node

Answer:

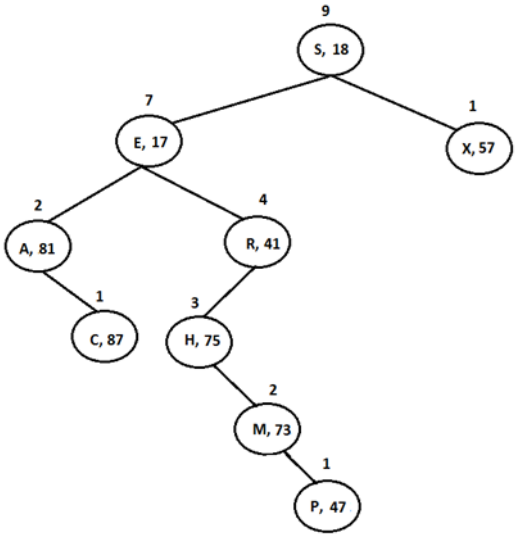
```
public void printPreorder() {
	printPreorder(root);
}

private void printPreorder(Node node) {
	if (node == null) return;
	System.out.print(node.key + ", ");
	printPreorder(node.left);
	printPreorder(node.right);
}

public void printPostorder() {
	printPostorder(root);
}

private void printPostorder(Node node) {
	if (node == null) return;
	printPostorder(node.left);
	printPostorder(node.right);
	System.out.print(node.key + ", ");
}
```

7. Provide the results of printInOrder, printPreOrder, and printPostOrder from the given BST.



Answer:

In-Order: **A, C, E, H, M, P, R, S, X**

Pre-order: **S, E, A, C, R, H, M, P, X**

Post-order: **C, A, P, M, H, R, E, X, S**