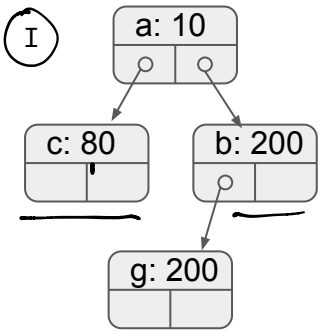
```
class Node<K,V> {
  K key; V value;
  Node<K,V> left, right;
  public Node(K key, V value,
              Node<K,V> left, Node<K,V> right) {
    this.key = key;
    this.value = value;
    this.left = left;
    this.right = right;
  }
}
```

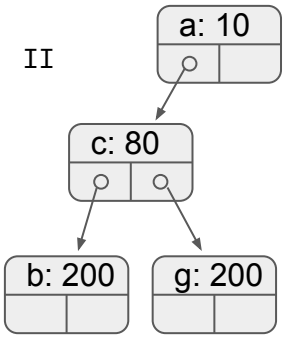
children

```
Node<String, Integer> node1 =
  new Node<>("a", 10,
    new Node<>("c", 80,
      new Node<>("b", 200, null, null),
      new Node<>("g", 200, null, null)),
    null);

Node<String, Integer> node2 =
  new Node<>("a", 10,
    null,
    new Node<>("c", 80,
      new Node<>("b", 200, null, null),
      new Node<>("g", 200, null, null)));

Node<String, Integer> node3 =
  new Node<>("a", 10,
    new Node<>("c", 80,
      new Node<>("b", 200, null, null),
      null),
    new Node<>("g", 200, null, null));

// Fill in the definition for the missing one
Node<String, Integer> node4 =
```

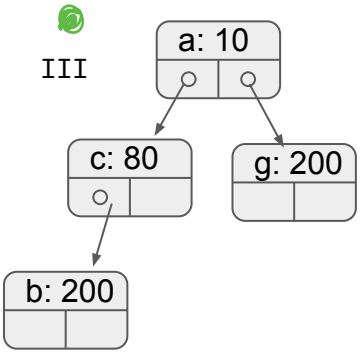
I



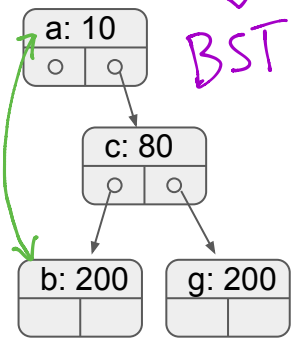
II



III



IV



BST

Which **tree** on the right is **NOT** represented by one of node1, node2, node3?

A: I B: II C: III D: IV
E. More than one of them is not represented

```
class Tree<K,V> {
  Node<K,V> root;
  Tree() { this.root = null; }
  Tree(Node<K,V> root) { this.root = root; }

  int countNodes(Node<K,V> node) {   // private helper




  }

  int countNodes() {     //public interface

  }

  V get(Node<K,V> node, K key) {




  }

  V get(K key) {


  }
}
```

Definition: A **binary search tree (BST)** is a tree where at **every** node, all keys to the **left** of that node are **smaller** than that key, and all keys to the **right** are larger.

Which **tree** on the front is a **binary search tree**?

A: I B: II C: III D: IV E: More than one of them is a BST

```
class BST<K,V> {
  Node<K,V> root;
  BST() { this.root = null; }
  BST(Node<K,V> root) { this.root = root; }
  V get(Node<K,V> node, K key) {
      if(node == null) { error }
      if(node.key.equals(key)) { return node.value; }
      if(node.key > key) { return get(node.left, key); }
      else { return get(node.right,key); }
  }
  V get(K key) {
    return this.get(root, key);
  }
  void set(K key) {
  }
}
```

PA7 - out this evening, due Mon, March 4

- closed
- builds on lecture today, mon + PA6

Exam next Wed (in class)

- logistics post coming
- cumulative; main topics $O/\Theta/\Omega$, sorting, hash maps