Discussion Section 9 (10/30/2008)

Tree

```
public abstract class Data {
   public abstract int compareTo(Data data);
   // -1 if this less than data; 0 if this equals data; 1 otherwise
}
interface Visitor {
   public void visit(Data data);
}
```

1.

```
class Tree1 {
   public Data data;  public List<Tree1> children;
   public Tree1(Data data) {
      this.data = data;  children = new List<Tree>();
   }
   public void addChild(Tree1 tree) {
      children.add(tree);
   }
   public void preorder(Visitor visitor) { // Implement




   }
   public boolean contains(Data data) { // Implement




   }
}
```

2.

```
class MyList<T> {  T head;  MyList tail;  }
class Tree2 {
   public Data data;  public MyList<Tree2> children;
   public Tree2(Data data) { // Implement

   }
   public void addChild(Tree2 tree) { // Implement

   }
   public void preorder(Visitor visitor) { // Implement




   }
   public boolean contains(Data data) { // Implement




   }
}
```

Heap (tree where a node is always greater than or equal to its children)

1.
```
class Heap1 extends Tree1 {
   public void addChild(Tree1 tree) { // Implement









   }
   // Extra fields or methods




}
```

2.
```
class Heap2 extends Tree2 {
   public void addChild(Tree2 tree) { // Implement









   }
   // Extra fields or methods




}
```

3. Exercise: How to implement node removal?