Free Response:

1. Consider the following interface `CityInfo` that will be used to represent cities in the United States. Each city is represented by its name and the name of the state in which it is located.

```
public interface CityInfo
{
  String city();
  String state();
}
```

The following class, `States`, will be used to store states and their respective cities. Information from `CityInfo` objects will be stored in this class as a `TreeMap`. In the `TreeMap`, the keys are the state names, and for each key the corresponding value is a `Set` of the cities in that state.

```
public class States
{
  private Map<String, Set<String>> theMap;

  public States() { theMap = new TreeMap<String, Set<String>>(); }

  // postcondition: Information from theCity
  // has been added to the Map
  public void addCityToMap(CityInfo theCity)
  { /* to be implemented in part (a) */ }

  public void printOneState(String theState)
  { /* to be implemented in part (b) */ }

  public void printAllStates()
  { /* to be implemented in part (c) */ }

  // ... other methods not shown
}
```

For example, assume that a `States` object, `stateMap`, has been initialized with the following `CityInfo` objects.

```
[Albany,NY] [Miami,FL] [Hamilton,NY]
[Jacksonville,FL] [Dallas,TX]
```

The following table represents the entries in `stateMap`.

Key	Value
FL	[Jacksonville, Miami]
NY	[Albany, Hamilton]
TX	[Dallas]

(a) Write the `States` method `addCityToMap`, which is described as follows. Method `addCityToMap` takes one parameter: a new `CityInfo` object, and updates `theMap` to include the information encapsulated in the `CityInfo` object. Method `addCityToMap` should run in $O(\log n)$ expected time where n is the number of states in `theMap`.

The following tables show the result of two sequential calls to `addCityToMap`, when applied to the object `stateMap` shown at the beginning of the question. Assume that `city1` has been defined as the `CityInfo` object `[Albany,GA]` and `city2` has been defined as the `CityInfo` object `[Houston,TX]`.

Result of the call

```
stateMap.addCityToMap(city1);
```

Key	Value
FL	[Jacksonville, Miami]
GA	[Albany]
NY	[Hamilton, Albany]
TX	[Dallas]

Result of the call

```
stateMap.addCityToMap(city2);
```

Key	Value
FL	[Jacksonville, Miami]
GA	[Albany]
NY	[Hamilton, Albany]
TX	[Dallas, Houston]

You are to complete method `addCityToMap` whose header is given below.

```
// postcondition: information from theCity
// has been added to theMap
public void addCityToMap(CityInfo theCity)
```

(b) Write method `printOneState`, which is described as follows. Method `printOneState` takes a `String` representing a state that is in `theMap`. It prints the name of the state and a list of cities in the state. The output should not include `[]`, and the cities should each be separated by a blank space.

For example, if `stateMap` contains the entries shown at the beginning of the question, the call `stateMap.printOneState("FL")` will result in the following output.

```
FL Jacksonville Miami
```

A solution that creates an unnecessary instance of any `Collection` class will not receive full credit.

```
public void printOneState(String theState)
```

c) Write method `printAllStates`, which is described as follows. Method `printAllStates` outputs the cities in each state in the format shown in part (b). The states should be listed in alphabetical order.

For example, if the `States` object `stateMap` has the following entries,

Key	Value
FL	[Jacksonville, Miami]
GA	[Albany]
NY	[Hamilton, Albany]
TX	[Dallas, Houston]

then the call `stateMap.printAllStates()` will produce the following output.

```
FL Miami Jacksonville
GA Albany
NY Albany Hamilton
TX Dallas Houston
```

In writing `printAllStates`, you may call `printOneState` as specified in part (b). Solutions that reimplement functionality provided by this method, rather than invoking this method, will not receive full credit. A solution that creates an unnecessary instance of any Collection class will not receive full credit.

```
public void printAllStates()
```