```
// Bellevue College CS211
// Fall 2016, Exam 1 (100 pts)
// W.P. Iverson, instructor          NAME: Bill Iverson

public static void main(String[] args) {
// Lines 1 to 8, you indicate the output, these are 5 points each.
                                                      x - Exam B
String[] words = {"zero", "one", "two", "three", "four"};
Map<String, Integer> data = new TreeMap<String, Integer>();
for (int i=0; i<words.length; i++) data.put(words[i], i);
// Above builds a Map that hopefully is obvious {e.g. … zero=0, …}

System.out.println(words.length);                      // 1. 5          4
System.out.println(data.size());                       // 2. 5          4

System.out.println(data.containsKey("four"));          // 3. true       F
System.out.println(data.containsValue(5));             // 4. false      F

System.out.println(data);  alphabetically       // 5. {four=4, one=1, three=3, two=2, zero=0}
System.out.println(data.get("two"));            // 6. 2                 1

Integer mystery = new Integer(42);
Integer secret  = new Integer(24);

System.out.println(42 < 24);                     // 7. false           T
System.out.println(mystery.compareTo(secret));   // 8. +1              -1


// The following code runs only after you complete items 9-14 (10 points each)

// We'll build a couple of objects, called first and second below
// Both Classes for these objects are incomplete (see attached)
// You get to complete the Class programming here to follow directions below

// Please write constructors that load the data from above upon instantiation
HASTreeMap first = new HASTreeMap(data);
// 9. Write this constructor on attached page        on attached
//         Be certain to included code to add these data (from array) into the Map

ISTreeMap second = new ISTreeMap(data);
// 10. Write this constructor on attached page, add data into Map

// In problems 11. and 12., you provide toString() code, or inherited,
// and write that code on attached Class page.

System.out.println(first);     // 11. Output should be identical to #5 above not needed
System.out.println(second);    // 12. Output should be identical to #5 above on attached

// 13. Provide an equals method for HASTreeMap (as used below), that returns true when
// two objects have exactly the same size Map, and all pairs are exactly the same
System.out.println(first.equals(second));             // returns true given data above
System.out.println(second.equals(first));             // returns true given data above
second.put("four", 44);
System.out.println(first.equals(second)); // returns false now, they are not the same
System.out.println(second.equals(first)); // returns false now, they are not the same

// 14. Provide an equals method for ISTreeMap (as used below), returns true when the two
//    objects passed have exactly the same size Map, and all pairs are exactly the same

// In Summary:     9 and 10 require correct constructors
//                 11 and 12 require correct toString() methods
//                 13 and 14 require each Class to have a correct .equals method.
```

```java
// This class "is a" TreeMap via inheritance (import java.util.* so all works)

public class ISTreeMap extends TreeMap<String, Integer> {

    // #10
    public ISTreeMap(Map<String, Integer> init) {
            super(init);
    }

    // #11  nothing to write, toString inherited

    // #14
    public boolean equals(HASTreeMap first) {
          return (this.toString().equals(first.toString)));
    }    ALT: return (first.equals(this));

}
```

```java
// This class "has a" TreeMap

public class HASTreeMap {

      private TreeMap<String, Integer> data;
      //9
      public HASTreeMap(Map<String, Integer> init) {
            data = new TreeMap<String, Integer>(init);
      }

      // #12
      public String toString() {
            return data.toString();
      }

      // #13
      public boolean equals(ISTreeMap second) {
            return (this.toString().equals(second.toString());
}     }  ALT: return (data.equals(second));
```