Test 1 Review: CompSci 6

Name (print): ____________________________

Honor Acknowledgment (signature): ____________________________

DO NOT SPEND MORE THAN 15 MINUTES ON ANY OF THE QUESTIONS! If you do not see the solution to a problem right away, move on to another problem and come back to it later.

Before starting, make sure your test contains 10 pages.

If you think there is a syntax error, then ask.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Value</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>6 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>8 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>15 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>21 pts.</td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>50 pts.</td>
<td></td>
</tr>
</tbody>
</table>
Problem 1: (Where's the love? (6 points))

Write a function countLove that returns the number of times the word "love" appears in a particular string.

For example,

<table>
<thead>
<tr>
<th>Function call</th>
<th>Return Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountLove(&quot;&quot;)</td>
<td>0</td>
</tr>
<tr>
<td>CountLove(&quot;love is what lovers love&quot;)</td>
<td>3</td>
</tr>
<tr>
<td>CountLove(&quot;love&quot;)</td>
<td>1</td>
</tr>
</tbody>
</table>

Complete countLove below:

```java
public int countLove (String str)
// post: returns the number of times "love" appears in s
// returns 0 if love does not appear
{

```
**PROBLEM 2:**  *(Out of order: 8 pts)*

**PART A**

Write the function `indexOfOutOfOrder` whose header is given below. The function `indexOfOutOfOrder` is given a array of numbers that are supposed to be sorted in increasing order and returns the index of the first number that is out of order. If none of the numbers are out of order, it returns -1.

For example, if the array `b` is the array below. Then the call `indexOfOutOfOrder(b)` would return 3, since 7, which is in position 3 is the first number out of increasing order.

```
3 4 9 7 12 11 18 10
```

Complete `indexOfOutOfOrder` below.

```java
public int indexOfOutOfOrder (int[] values)
// pre: values.length > 0
{

}
```
PROBLEM 3:  *(Glory Days: (15 pts))*

Implement a class that represents a music jukebox. To support its use in as many diners as possible, your class should read its list of songs from a data file, one song per line. Each time the jukebox is requested to play a song, a random one from those read in should be returned.

For example, the code below represents how your class might be used in a program.

```java
public static void main (String args[]) {
    String songFile = "easy_listening.dat";
    Jukebox box = new Jukebox(new Scanner(new File(songFile)));

    while (box.numSongsPlayed() < 10) {
        cout << box.playSong() << endl;
    }
}
```

Below is some possible sample output from the program above given a data file that contains the kind of music I like to listen to:

How many dollars do you want to spend, between 1 and 10? 1
Captain & Tennille - Love Will Keep Us Together
Tom Jones - Its Not Unusual
Donna Summer - Dim All The Lights
Kool and The Gang - Ladies Night
Bee Gees - Night Fever
**Part A:** (3 points)
Complete the declaration of the class' instance variables below. You should add any private state (instance variables) you think your JukeBox class needs in order to implement the constructor and other two methods of the class so that they function as shown in the example above.

```java
public class JukeBox
{
    // Part A: declare instance variables here

    public JukeBox (Scanner scanner)
    {
        // implemented in Part B
    }

    public int numSongsPlayed ()
    {
        // implemented in Part B
    }

    public String playSong ()
    {
        // implemented in Part B
    }
}
```
**Part B:** (12 points)

Write the implementation for the class member functions previously declared below. Hint: use the Scanner to get an entire line of text from a file, store each line in a TreeSet, and then make use of the Random class to generate a random number to select an answer from the list.

Note, you may lose points on Part A for instance variables that are declared there, but not used in this Part.

```java
public JukeBox (Scanner scanner)
    // TODO: read all lines from file and store as possible songs
    {

    }

    public int numSongsPlayed ()
    // TODO: return number of times a song has been played
    {

    }

    public String playSong ()
    // TODO: return one random string from set of possible songs
    {

    }
```
**PROBLEM 4:**  (*Working Together (21 pts)*)

Assume that every player on a team is modelled using the class `Player` declared as shown below.

```java
class Player {  
    private String myName;  
    private int[] myMinutes; // minutes played in each game  

    public Player (String name, int[] minutes)  
    {  
        myName = name;  
        myMinutes = minutes;  
    }  

    public int getMinutesForGame (int whichGame)  
    // pre: 0 <= whichGame < myMinutes.length  
    {  
        return myMinutes[whichGame];  
    }  

    public int getNumberGamesPlayed ()  
    {  
        return myMinutes.length;  
    }  

    public String getName ()  
    {  
        return myName;  
    }  
}
```
Part A (10 points)
Write the function `readTeam` below that reads data from a file and stores it in an initially empty set of Player's. Assume that the text file with team information being read is in the format shown below. There are two lines for each player in the team. The first line contains first and last name, and the second line contains a list of the number of minutes played in each game during the season. Notice that the number of games played varies from player to player and thus each list is preceded by a count declaring the number of games the player played.

A sample data file is shown below with the first player with 4 games and the second play with 7 games:

Fred Smith
4 1 13 5 1
Chris Jones
7 8 2 5 6 4 7 16

Complete `readTeam` below.

```java
public TreeSet readTeam (Scanner scanner)
// pre: scanner is set to read file in the correct format
// post: reads information and stores it in team
{
```
Part B (4 points)
Write the function *totalNumberOfMinutes* below that, given a *Player*, returns the total number of minutes that player played during the season.
Given the example players in the previous part, your function should return the value 20 minutes when called with the *Player* object representing Fred Smith and 48 minutes when called with the *Player* object representing Chris Jones.
Complete *totalNumberOfMinutes* below.

```java
public int totalNumberOfMinutes (Player player) {
```

```java
}
```
Part C (7 points)

Write the function `mostMinutes` below that, given a set of players representing a team, returns the `Player` who has played the most total minutes during the season.

Given the example team from the previous parts, your function should return the `Player` object representing Chris Jones because he played for 48 total minutes, while his only other team member played for only 20 minutes.

You will not receive full credit for this part unless you call the function `totalNumberOfMinutes` that you wrote in Part B at least once and use its result in determining the result of this function. Assume `totalNumberOfMinutes` works as specified regardless of what you wrote in Part B.

Complete `mostMinutes` below.

```java
public Player mostMinutes (TreeSet team) {
```