PROBLEM 1:  \textit{(Loop de loop: (10 pts))}

Consider the following two solutions for removing odd numbers from an ArrayList of Integers. For example, assume the ArrayList values contains the numbers:

\begin{verbatim}
  3  5  6  24  7  9  1  8  28  11
\end{verbatim}

Then with the call to removeOddNumbers, the ArrayList returned should contain:

\begin{verbatim}
  6  24  8  28
\end{verbatim}

PART A (6 pts):
The following implementation of removeOddNumbers DOES NOT WORK correctly.

```java
public ArrayList<Integer> removeOddNumbers (ArrayList<Integer> values)
{
    for (int k=0; k < values.size(); k++)
        {
            if (values.get(k) % 2 == 1) // number is odd
                {
                    values.remove(k);
                }
    }
    return values;
}
```

a. Give the values in the ArrayList that is returned.

b. Explain why it does not work correctly.
PROBLEM 2:  (Too many interests: (18 pts))

PART A (8 pts):
Write the method `numberOccurrences` whose header is given below. This method has two parameters, a String called `name` and an ArrayList of type TreeSet called `sets`. (Each set will be a TreeSet of type String.) This method returns the number of sets that name appears in. For example, assume the sets in the ArrayList are the following:

Set 1: Hu, Lamela, Montgomery, Senko, Shearer  
Set 2: Wilde, Lim, Kurtzman, Kenney, Montgomery, Shearer  
Set 3: Scheerer, Cha, Wilde, Senko  
Set 4: Garrison, Kozikowski, Montgomery, Shearer  
Set 5: Pollack, Neeves, Montgomery, Shearer, Wilde

The call, `numberOccurences("Shearer", sets)` returns 5 (it is in all five sets), `numberOccurences("Senko", sets)` returns 2 (it occurs only in sets 1 and 3), and `numberOccurences("Fox", sets)` returns 0 (it is not in any of the sets).

Complete the method `numberOccurrences` below.

```java
// return the number of occurrences of item in sets.
int numberOccurrences(String name, ArrayList<TreeSet> sets)
{
```
PART B (10 pts):
Write the method `inTooManyClubs` whose header is given below. This method has one parameter: an `ArrayList` of type `TreeSet` named `membersets`. Each set represents the list of students in a club at Dook University. Dook has a rule that students can only be involved in a maximum of three clubs. This method returns an `ArrayList` of all the students who are in more than three clubs.

For example, if we consider the `ArrayList` of `TreeSets` called `sets` from Part A, then the call `inTooManyClubs(sets)` returns the `ArrayList` containing the names: Shearer and Montgomery, the only two who are in more than three clubs.

In writing `inTooManyClubs` you should call the method `numberOccurences` that you wrote in Part A. Assume that `numberOccurences` is correct, regardless of what you wrote.

Complete the method `inTooManyClubs` below.

```java
ArrayList<String> inTooManyClubs (ArrayList<TreeSet> membersets) {
    
    }
```
PROBLEM 3:  (This One’s a Little Loopy: (18 pts))

Part A: (8 points)
Write the method isAscending that, given a list of Comparable objects, returns true if all of the objects are in ascending order (i.e., each object is less than the next one), and false otherwise.
You will not receive full credit if you cast the objects in the given list to any type more specific than Comparable.
Complete the method isAscending below.

    public boolean isAscending (List<Comparable> data)
    {
    }

}
Part B: (10 points)
Write the function `minInARow` that given a list of numbers and a specific value returns the length of the shortest sequence of those values that appear consecutively.
For example, given the list below, the length of the shortest consecutive sequence of ones is 2, twos is 8, and fours is 1. If the value does not appear in the list, you should return 0.

```
1 1 2 2 2 2 2 2 2 1 1 1 4 1 1 4
```

You will not receive full credit if you cast the objects in the given list to any type more specific than `Object`.
Complete the method `minInARow` below.

```java
/**
 * returns length of shortest sequence of values that appear consecutively
 */
public int minInARow (List<Object> numbers, int value)
{
    
```
Students want to record ratings for each professor they have. Each professor is rated on a scale of 1-10 where 1 is terribly abominable and 10 is supremely wonderful. Write several methods to tabulate the results using the class `ProfRate` below which stores information about one professor’s rating and the class `Student` which stores one student’s ratings.

(The methods you write start on the next page.)

Each student’s ratings are stored in a map of strings, each professor’s name, to ints, their rating by that student. Ratings are stored with a student’s name so that each student can vote only once.

```java
public class Student {
    private String myName;
    private Map<String, Integer> myRatings;

    public Student (String name) {
        myName = name;
        myRatings = new TreeMap<String, Integer>();
    }

    public void addRating (String professor, int rating) {
        myRatings.put(professor, new Integer(rating));
    }

    public int getRating (String professor) {
        if (myRatings.containsKey(prof)) {
            return myRatings.get(prof);
        } else {
            return 0;
        }
    }
}
```
Part A (10 points)
Suppose each student’s ratings are stored in a file in the format below: the student’s name on one line
followed by several lines with each line containing a rating number followed by the professor’s name who
achieved that rating. There are as many lines after the student name as there are professors rated by the
student.

Joe Student
8 Michael Marxist
7 Nancy Naturalist
2 Henrietta Humanist

Complete the function readRatings below that reads such a file and returns a Student object containing
the information in the file.

Student readRatings (Scanner input)
{
    Student result = new Student(input.nextLine());

    // read student ratings

    return result;
}

return result;
}
Part B (10 points)

Write the method `averageRating`, that returns the average rating of a professor as rated by every student whose information is stored in a list. For example, if Professor Smith has ratings of 4, 4, 7, and 9 by four students, the average rating is 6.0. If a student does not rate a professor, the student should not affect the professor’s average rating.

```java
double averageRating (List<Student> students, String professor)
{
```
Part B (10 points)

Write the method `highestRating`, that, given a list of professor names and a list of students, returns the name of the professor with the highest average rating by the students.

```java
String highestRated(List<String> professors, List<Student> students) {
}
```
PROBLEM 2: (How to choose: (22 pts))

Part A: (10 points)
Consider the problem of building a registration program for a university that manages students wanting to register for courses, taking courses, and receiving grades for those courses.
For each of the following problems, state the type of collection that best solves it and justify your decision. If you are using a map, you must clearly state what are the keys and values.

- a course contains a collection of enrolled students
- a course contains a collection of students on a waitlist to enroll
- a course contains a collection of student grades
- a course contains a collection of labs in which students are enrolled
- a student’s schedule contains a collection of course times in which the student is enrolled (you may assume all times are regular and do not overlap)
Part B: (12 points)
For each of the following problems, state the type of collection that best solves it and give pseudocode to solve the problem using that collection (do not write Java code). If you are using a map, you must clearly state what are the keys and values.

- **Random Phone Numbers** given a input file representing all area codes in the United States and a number $N$, generate $N$ unique random phone numbers (i.e., a valid area code followed by seven digits)

- **Spell checking** given a input file representing a dictionary of words, an input file of the user’s personal dictionary, and an input file of text, report those words in the text file that are not spelled correctly according to the words in any of the dictionaries

- **Spell correcting** given a input file representing a dictionary of words with common mis-spellings for each of those words and an input file of text, report those words in the text file that are not spelled correctly according to the words in the dictionary as well as suggested corrections