PROBLEM 1 : (*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:

```
3  5  6  24  7  9  1  8  28  11
```

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

```
6  24  8  28
```

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

```
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.

PART B ***(4 pts)***:

Here is another implementation of removeOddNumbers that works correctly.

```
public ArrayList<Integer> removeOddNumbers (ArrayList<Integer> values)
{
  ArrayList<Integer> temp = new ArrayList<Integer>();

  for (int k=0; k < values.size(); k++)
  {
     if (values.get(k) % 2 == 0) // number is even
     {
        temp.add(values.get(k));
     }
  }
  values.clear();

  for (int k=0; k<temp.size(); k++)
  {
     values.add(temp.get(k));
  }
  return values;
}
```

Give a meaningful loop invariant for the first for loop.

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: Schearer, 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 it sets 1 and 3), and `numberOccurences("Fox", sets)` returns 0 (it is not in any of the sets).

Complete the method `numberOccurences` below.

```
// return the number of occurences 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.

```
ArrayList<String> inTooManyClubs (ArrayList<TreeSet> membersets)
   {




}
```

PROBLEM 3 : (*Put Prof. Rodger Behind Bars* (10 pts))

Write the `execute` method of `BarsOn` whose header is given below. This method takes a color Pixmap image and adds vertical black bars to it. Each black bar is 10 columns in width, the first one starts in column 0 and a new bar should be drawn every 40 columns (the second bar starts in column 40). A partial bar may be drawn as the rightmost bar if there are not 10 columns for that bar. Note that the Color black is when red, green and blue are all set to 0.

For example, in the figure below, a color picture of Prof. Rodger is shown on the left and the same image is shown on the right after the BarsOn button has been pressed.

```
public class BarsOn extends Command {
    public BarsOn () {
        super("BarsOn");
    }

    public void execute (Pixmap target) {
        Dimension bounds = target.getSize();









    }
}
```

PROBLEM 4 : (*Generating questions* (36 pts))

Consider the following abstract Quiz class for creating a quiz.

```
public abstract class Quiz {
   private int score;              // score for Quiz
   private String myQuestion;      // current question
   private String myAnswer;        // current answer to question
   protected Random myGenerator;  // random number generator

   public Quiz (){
      score = 0;
      myGenerator = new Random();
      myQuestion = "DEFAULT question";
      myAnswer = "DEFAULT answer";
   }

   abstract public void createQuestion();

   public void runQuiz(int numQuestions)
   {
      Scanner console = new Scanner(System.in);
      System.out.println("This quiz will have " + numQuestions + " questions");

      for (int k=1; k<= numQuestions; k++)
      {
         createQuestion();
         askQuestion();

         String userAnswer = console.next();
         if (userAnswer.equals(getCorrectAnswer()))
         {
            score ++;
            System.out.println("Correct, current score is: " + score);
         }
         else
         {
            System.out.println("Answer is incorrect. ");
            System.out.println("The correct answer is: " + getCorrectAnswer());
         }
      }
      System.out.println("Quiz is over, your score is: " + score + "/" + numQuestions);
   }
```

```
   public String getCorrectAnswer()
   {
      return myAnswer;
   }
   public void askQuestion()
   {
      System.out.println(myQuestion);
   }
   protected void setQuestion(String question)
   {
      myQuestion = question;
   }
   protected void setCorrectAnswer(String answer)
   {
      myAnswer = answer;
   }
}
```

PART A ***(10 pts)***:

1. Explain why there is no code for the method *createQuestion*.

2. What is the output when the person taking the quiz gives a wrong answer?

3. Suppose someone wants to run a quiz and enters the following code into a Main method.

```
Quiz q = new Quiz();
q.runQuiz(4);
```

   If this code is valid, explain what happens. If it is not valid, explain why.

PART B *(10 pts)*:

We would like to create a MathQuiz for randomly generating addition and multiplication questions for pairs of numbers in the range from 0 to 99. For example, it might randomly generate the question "4 * 76 =", or the question "84 + 35 =" .

The class MathQuiz has been started below for you. You should complete the MathQuiz method `createQuestion`. This method should generate a math question and a correct answer and store them in the Quiz state. The math question should randomly be either an addition or a multiplication question involving two integers in the range from 0 to 99.

```
public class MathQuiz extends Quiz
{
  // NO State needed, use the state in the Quiz class
   public MathQuiz()
   {
      super();
   }

   public void createQuestion()
   {









   }
}
```

PART C ***(16 pts)***: We would now like to generate a History quiz that asks questions from a file that have one word answers (also from a file).

For example, a question might be "Who was the only female president at Duke University?" and the answer would be "Keohane."

```
public class HistoryQuiz extends Quiz
{
   private ArrayList<String> myQuestions;  // store questions
   private ArrayList<String> myAnswers;    // store one word answers

   public HistoryQuiz(String filename)
   {
      // Code not shown
   }

   public void createQuestion()
   {
       // Code not shown
   }
}
```

PART C.1 ***(10 pts)***:

Complete the constructor for the HistoryQuiz **on the next page**. You should read from a file that has a series of questions and answers. A question appears on one line and the one word answer follows immediately on the next line. Store the questions and answers in the parallel arrays myQuestions and myAnswers so they can be used later to generate questions and answers.

A sample file might be:

```
Who was the only female president at Duke University?
Keohane
Who was the first president of the United States?
Washington
What year did Christopher Colombus discover America?
1492
What number president of the US was Harry Truman?
33
```

Complete the code for the HistoryQuiz constructor that has been started below.

```
   public HistoryQuiz(String filename)
   {




     try
      {
```

```
      }
       catch (IOException e)
       {
           System.out.println("Error reading file " + filename);
           System.out.println(e);
       }
}
```

PART C.2 ***(6 pts)***:

Complete the HistoryQuiz method `createQuestion`. This method should randomly pick a question from the stored questions and set the question and correctAnswer in the Quiz state.

Complete `createQuestion` below.

```
public void createQuestion()
{









}
```

```
public class String
{
    // Returns the length of this string.
    public int length ()

    // Returns a new string that is a substring
    // of this string. The substring begins at
    // the specified beginIndex and extends to
    // the character at index endIndex - 1.
    public String substring (int beginIndex,
                             int endIndex)

    // Returns the index within this string of
    // the first occurrence of str
    // returns -1 if str is not found
    public int indexOf (String str)

    // Returns the index within this string of the
    // first occurrence of str after index start
    // returns -1 if str is not found
    public int indexOf (String str, int start)
}

public class Pixmap
{
  // create a new Pixmap
  public Pixmap(int width, int height)

  // returns true if (x,y) is in the bounds of the Pixmap
  public boolean isInBounds(int x, int y)

  // return the Dimension of the pixmap, Dimension has a width and height
  public Dimension getSize ()

  // returns the Color of the pixel at (x,y)
  public Color getColor (int x, int y)

  // sets the Color of the pixel at (x,y)
  public void setColor (int x, int y, Color value)

}

public class Color
{
   // create a Color with each of r, g, b in the range from 0 to 255
   public Color(int r, int g, int b)
```

```
   // returns the blue color (0 to 255 value), similar methods for red and green
   public int getBlue()
}

public class Dimension
{
    // returns height of Dimension
    public int getHeight()

    // returns width of Dimension
    public int getWidth()
}

public class ArrayList
{
    // Constructs an empty list
    public ArrayList ()

    // Returns the number of elements in this list.
    public int size ()

    // Returns element at index in this list.
    public Object get (int index)

    // Replaces the element at the specified position
    // in this list with the specified element.
    public Object set (int index, Object element)

    // Appends specified element to end of this list.
    public boolean add (Object o)
}


public class File
{
  // Open a new file from the given pathname
  public File (String pathname);
}

public class Scanner
{
    // Create Scanner that reads data from a file.
    public Scanner (File file)

    // Create Scanner that reads data from a string.
    public Scanner (String str)
```

```
    // Change delimiters used to separate items
    public void useDelimiter (String characters)

    // Check if more items are available
    public boolean hasNext ()

    // Get next delimited item as a string
    public String next ()

    // Get next line as a string
    public String nextLine ()

    // Get next delimited item as an integer value
    public int nextInt ()

    // Get next delimited item as a Double value
    public int nextDouble ()
}

public class TreeSet
{
   // creates an empty TreeSet
   public TreeSet()

   // adds object e to the TreeSet
   boolean add(Object e)

   // removes all objects from the TreeSet
   void clear()

   // returns true if e is in the set, othewise returns false
   boolean contains(Object e)

  // returns true if set is empty, otherwise returns false
  boolean isempty()

  // returns an Iterator for the set
  Iterator<Object> iterator()

  // removes the object e from the set
  boolean remove(Object e)

  // returns the number of elements in the set
  int size()

}
```

```
public class Random
{
  // constructor
  public Random ()

  // returns random number from 0 (inclusive) to n (exclusive) (does not
  // include n)
  public int nextInt(n)
}
```