**ITEC 2150 Final Exam**

**Summer 2016**

Below are 4 questions. To receive full credit, you must answer 3 of them. If you do all 4 of them, I will grade them all and count the highest 3 for your final. The 4th will count as a bonus worth up to 20 points on your final. You will have 2.5 hours to complete the final.

IF YOU USE ANY OTHER FILES BESIDES THE ONES LISTED, YOU MUST SUBMIT THEM. ANY FILE NEEDED TO RUN YOUR PROGRAM SHOULD BE SUBMITTED.

# File/IO

1. Create a program that prints out the last five lines in a text file to the console.

For example, a text file may contain the following text:

1. All work and no play makes Jill a dull girl

2. All work and no play makes Jack a dull boy

3. All work and no play makes Jill a dull girl

4. All work and no play makes Jack a dull boy

5. All work and no play makes Jill a dull girl

6. All work and no play makes Jack a dull boy

7. All work and no play makes Jill a dull girl

8. All work and no play makes Jack a dull boy

9. All work and no play makes Jill a dull girl

10. All work and no play makes Jack a dull boy

Your program will print the following:

6. All work and no play makes Jack a dull boy

7. All work and no play makes Jill a dull girl

8. All work and no play makes Jack a dull boy

9. All work and no play makes Jill a dull girl

10. All work and no play makes Jack a dull boy

Notice that the last **five** lines are printed. The most efficient way to solve this problem is to keep track of fives lines of text in an ArrayList (or array) and print it out once you reach the end of the file. You may use any other approach to solve this problem but it will result in a deduction of 10 points. Also, you cannot store all the text in the ArrayList or read the text twice (-10 point deduction for each).

Please turn in the file “Tail.java” which contains your solution.

# Interfaces

1. Modify the given Customer.java to implement the Comparable interface so that the customer will be sorted by their last name in ascending order. Implement a toString() method for the Customer class. Using the CustomerTester.java, add a least 5 new Customers to an ArrayList. Print the unsorted list. Sort the list of customers, and then print out the sorted ArrayList of Customers. Make sure your output is meaningful and readable.  
     
   Submit Customer.java and CustomerTester.java.  
     
   Sample output:  
   Customer 1 is Tugg Thomson  
   Customer 2 is Fred Jones  
   Customer 3 is Sally Stevens  
   Customer 4 is Tim James  
   Customer 5 is George Thompson  
     
     
   Customer 1 is Tim James  
   Customer 2 is Fred Jones  
   Customer 3 is Sally Stevens  
   Customer 4 is George Thompson  
   Customer 5 is Tugg Thomson

# Recursion

1. Modify the given file Sixes.java to return the number of sixes contained in the numbers supplied. This must be a recursive solution. Non-recursive solutions will **not** receive credit. Remember, to be a recursive solution, your method must call itself. You should not have a loop anywhere in this program. Submit your updated Sixes.java file.  
     
   Sample output:

66776 has 3 sixes.

123456 has 1 sixes.

111223 has 0 sixes.

666666 has 6 sixes.

868686838 has 3 sixes.

6241 has 1 sixes.

776 has 1 sixes.

# Polymorphism

1. Sally runs a bike store but has a problem with her suppliers. She does not know what type of bike she will receive until it arrives. She only knows she will receive 10 bikes every day. She wants to be able to create a list of bikes as they are received and display this on her computer so she will know what types of bikes arrived. Fred agreed to help her by writing a Java program that would allow this to be automated. Unfortunately, a new bike arrived and Fred has been off riding and has not been seen since.  
     
   Sally sells three types of bikes. A regular bicycle with a max speed of 20 mph. It is powered by pedaling. A motorcycle with a max speed of 100 mph that is powered by gasoline. Finally, a scooter with a max speed of 30 mph that is powered by electricity.   
     
   Fred has created a base class, Bike.java. Sally does not want us to modify this since Fred may return from his ride and she doesn’t want to upset him  
     
   Your job is to create three classes that inherit from Bike and a BikeTester class. The three classes are Bicycle.java, Motorcycle.java and Scooter.java. Each of these classes will need to complete the implementation began in Bike.java. Since Sally doesn’t want us to change Fred’s code, you will need to create instance variables in each of these classes to hold the max speed and the power type. Notice Fred has plans for a custom exception to be thrown from Bike.java. We will need to create this exception.  
     
   Finally, create a class that will randomly generate the bikes Sally receives in a day. This class should create the appropriate number of bikes of random types. Once these are created, you should use the toString() to print the list of bikes. The normal ratio of bikes received is 3 Motorcycles, 3 Scooters and 4 Bicycles. You should build your program to use a random distribution based on this normal distribution. Submit files Bicycle.java, Motorcycle.java, Scooter.java, BikeTester.java and UnkownPowerTypeException.java.  
     
   Sample output:

Your bike is a Scooter with a max speed of 30

It is powered by electric.

Your bike is a Scooter with a max speed of 30

It is powered by electric.

Your bike is a Scooter with a max speed of 30

It is powered by electric.

Your bike is a Bicycle with a max speed of 20

It is powered by pedal.

Your bike is a Scooter with a max speed of 30

It is powered by electric.

Your bike is a Scooter with a max speed of 30

It is powered by electric.

Your bike is a Scooter with a max speed of 30

It is powered by electric.

Your bike is a Scooter with a max speed of 30

It is powered by electric.

Your bike is a Motorcycle with a max speed of 100

It is powered by gas.

Your bike is a Scooter with a max speed of 30

It is powered by electric.