CSCE 747 - Assignment 5: Remaining Topics

Due Date: Monday, April 25, 11:59 PM

There are 7 questions, worth a total of 100 points. You may discuss these problems in your teams and turn in a single submission for the team on Moodle. Answers must be original and not copied from online sources.

Problem 1 (15 Points)

Complete exercises 18.3 from the textbook. Explain your answers.

Problem 2 (30 Points)

Organizations that make use of the extreme programming (XP) development ideology utilize a "test first" approach when developing software – they write all unit tests for a class before actually creating the code for that class.

As a developer working on the next generation of CoffeeMaker, you are tasked with adding a new "money manager" class. Now, when a drink is requested, the user must insert money (the hardware mechanism for this is outside of scope – you just need to perform software checks on the amount inserted). Your new "money manager" class must perform the following tasks:

- Check whether the amount inserted matches the amount of money required for the requested recipe.
- If more money was inserted than was required, return the remainder to the user.
- Track the amount of money spent on each type of drink since the CoffeeMaker was turned on (since the program began execution – you do not need to keep a permanent record).

Note that some of this functionality is currently implemented in the CoffeeMaker class and the main interface. In your refactored system, that functionality should instead be offered by the money manager.

1. Design this class using the "test first" development paradigm:
    a. Write a class outline – include a list of data members and methods needed to perform the above tasks, with descriptions for each.
    b. Write unit tests for the class and its methods to ensure that it performs all required functions correctly.
    (20 Points)
    You do NOT need to actually program the class.
2. An ideal, but usually unachievable, testing goal is to automatically derive test oracles from the same specification statement that establishes the acceptable behavior of a program or module. To what extent does the "test first" approach of extreme

programming achieve this goal? Discuss advantages and limitations of using test cases as a specification of behavior. (10 Points)

Problem 3 (20 Points)

Complete exercises 4.2 and 4.3 from the textbook. Explain your answers.

Problem 4 (20 Points)

You are testing a gadget - the “new new thing” - for your very own startup. You know your competition just released their product, but they have gotten really bad press because of its poor quality. You want to wow your customers (and investors) with a cool gadget that actually works. From your competitor’s experience (you have been following their product release very closely), you know the customer wants an **availability** of at least 99% and a **rate of fault occurrence** of less than 2 failures per 8-hour work period.

After the last fixes to your code, you have been testing your product for exactly 14 full (24 hour) days. The product failed a total of 45 times during this period and it took an average of 18 minutes to restart the product after each failure (mainly because of the difficulty of getting to the very small and poorly placed hardware reset button - it is placed on the back of your very heavy gadget and your gadget is always placed under a desk).

You are about to make a critical presentation to your investors. They want to know the measured availability and rate of fault occurrence, and they want to know if you can start shipping. If you cannot start shipping, they also want to get an idea of how the problem can be addressed.

1. What is the rate of fault occurrence?
2. What is the availability?
3. Is the product ready to ship?
4. If not, why not? Any suggestions on how to fix it?

Problem 5 (15 Points)

Complete exercises 22.5 from the textbook. Explain your answers.