CMSC 132 Quiz 5 Worksheet

The following list provides more information about the quiz.

- ≅ The quiz will be a written quiz (no computer).
- ≅ The quiz will be in lab/discussion session.
- ≅ Closed book, closed notes quiz.
- ≅ Answers must be neat and legible. **You must use pencil.**

1. What are two advantages to multi-threading?
2. What are two disadvantages to using multi-threading?
3. What are two ways to create threads in Java?
4. What is a data race? How can you avoid it?
5. Give an example of a Java code with a data race.
   a. Eliminate the data race using synchronized methods, e.g., synchronized foo( ) { … }
   b. Eliminate the data race using synchronized objects, e.g., synchronized(bar) { … }
6. Modify the following class so we can create threads that print messages. For the modified class, provide a main method that creates and starts two threads, one printing the message "Testudo" and the other the message "Terps". The main thread will display "UMCP" after the previous two threads have finished.

```
public class PrtMessage  {
    private String message;

    public PrtMessage(String message) {
            this.message = message;
    }

    public void print() {
            for (int i=0; i<50; i++)
                    System.out.println(message);
    }

}
```

7. The following class implements a model of a student dining hall serving pizzas to students. 10 pizzas are baked, then served to 20 students. Students are numbered between 0 and 19 in the order they are served. A message is printed indicating whether a student starved or was served a pizza.

   a. Rewrite the DiningHall class so that after the makePizza( ) method is called 10 times, the servePizza( ) method is called once each from 20 different threads.
   b. Insert synchronization to eliminate data races in your code, if any exist.
   c. Describe what data races may occur in your multithreaded code without synchronization.

```
public class DiningHall {
    static int pizzaNum;
    static int studentID;
    public void makePizza() { pizzaNum++; }
    public void servePizza() {
            String result;
            if (pizzaNum > 0) { result = "Served "; pizzaNum--; }
            else result = "Starved ";
            System.out.println(result + studentID);
            studentID++;
    }
    public static void main(String[] args) {
            DiningHall d = new DiningHall();
            for (int i = 0; i < 10; i++)
                    d.makePizza();
            for (int i = 0; i < 20; i++)
                    d.servePizza();
    }
```

```
}
```

8. Modify the **Product** class so we can compute the product of the elements of the **data** array using two threads (in addition to the main thread). One thread should compute the product of the first half of the array and the second one will take care of the rest. Make sure your code has no data races.

```
public class Product {
    public int[] data;

    public Product(int[] data) {
        this.data = data;
    }

    public int computeProduct(int startIndex, int endIndex) {
        int result = 1;
        for (int i = startIndex; i <= endIndex; i++) {
            result *= data[i];
        }
        return result;
    }

    public static void main(String[] args) {
        int[] data = {2, 3, 4, 2, 2, 3};

        Product product = new Product(data);
        System.out.println(product.computeProduct(0, 5));
    }
}
```