SFDV3006 Concurrent Programming
Lab 4 – Semaphores, synchronization and condition synchronization

Problems:

1. An object of class *ResourceControl* allows access to three shared resources from multiple threads. A thread which wants to use a resource calls *acquire()*, after it finishes using the resource it calls *release()* to release the resource.
   ResourceControl should block a thread if there are no resources available. Also it must notify another thread when a resource becomes available.
   Write the *ResourceControl* using a Semaphore to control access to the three shared resources.

2. How would you implement the *ResourceControl* class without using Semaphores but by using condition variables? Write the Java code for your implementation.

3. Implement the *SingleSlotBuffer* from Lab 3 using semaphores only for both synchronization and condition synchronization. Your solution must be free from deadlocks and other problems such as livelocks and starvation.

4. Implement a *BoundedCounter* which has two methods *increment()* and *decrement()*. *increment()* will increment the counter value by 1 and *decrement()* will decrease it by 1. However the *increment()* should not increment the value more than a maximum of 10 and decrement should never reduce by 0.
   Assume that different increment and decrement threads call the increment and decrement methods many times.

5. A class *LogWriter*, which writes messages to a log file is as shown below:

```
import java.io.*;
class LogWriter {
   public static void log(String logmsg) throws IOException{
      FileWriter fw = new FileWriter("log.txt");
      fw.write(logmsg);
      fw.close();
   }
}
```

   The above class is not thread safe. Rewrite the *LogWriter* class to use a *Semaphore* for mutual exclusion such that only one thread can execute the *log()* method

6. The above *LogWriter* class is an example of a Write-Only database. Assume that the Write Only Database has the following methods:
   startWrite()
   endWrite()
   How would you implement these methods so that only one thread in allowed to write to the shared file /database? Write a *WriteOnlyDB* Java class as monitor with *startWrite()* and *endWrite()* methods with the correct condition synchronization using semaphores.