CSC 227 Operating Systems Assignment 2
Due date: 28-March-2021

Instructions:
This is an individual assignment.
The homework must be **submitted electronically through LMS**

Part 1: Thread creation (30%)

a. Using Java multithreading library, write a Java program that calculates the sum of the numbers from 1 to 100,000,000. Split the numbers between four threads equally where each thread calculates the sum of one fourth of the numbers. For example, the 1st thread will calculate the sum of the numbers from 1 to 25,000,000 whereas the 2nd thread will calculate the sum of the numbers from 25,000,001 to 50,000,000 and so forth. The main thread will have to print out the sum after gathering the results. Note that you have the choice to create threads by either implementing Runnable interface or extending Thread class.

b. Now, write a sequential version of the program described above using a single main thread (i.e., without multithreading). Make sure to record and print out the time spent during the execution of both sequential and multithreaded versions (hint: you may consider using **`System.currentTimeMillis()`** to record execution time).

Part 2: Synchronization (40%)

Assume that we have a file named "sharable.txt" which can be shared among several threads. We want to write a program that controls the access to that file in a way that only one thread at a time is allowed to access it (i.e., for writing/appending purposes). Using Java multithreading, write a program that creates three threads and assigns a number to each thread. Then, each thread will start running by executing a code for opening the file "sharable.txt" and writing the following lines:

```
Thread x started writing
Thread x is currently writing
Thread x finished writing – Student Name
```

Your program should allow only one thread -at any given time-to access the file and write in it. It also should keep away any thread from overwriting the lines written by any other threads (hint: use synchronized methods/blocks). Finally, when the execution of your program is completed, the output stored in "sharable.txt" should look like the following – put your name in place "**`Your Name`**":

```
Thread x started writing
Thread x is currently writing
Thread x finished writing – Your Name
Thread y started writing
Thread y is currently writing
Thread y finished writing – Your Name
Thread z started writing
Thread z is currently writing
Thread z finished writing - Your Name
```

Part 3: Interrupt handling (30%)
In Java multithreading environment, one thread can send an interrupt to another by calling the **`interrupt()`** method on the Thread object for the target thread (i.e., the thread to be interrupted). To handle interrupts in a target thread, Java allows two approaches. One is performed by writing an exception handler for **`InterruptedException`** (only applicable if the target thread is invoking methods which throw that exception such as **`sleep`**). The other approach is performed by periodically checking the interrupt status flag **`Thread.interrupted`** and performing the handling routine when that flag is set to true.

Write a Java program that illustrates the use of the two approaches described above. Your program should start by creating two threads, each thread should use different interrupt handling approach. Then, the program needs to send interrupts to each one of the created threads such that a thread needs to return (i.e., stop execution) after receiving an interrupt from the main thread. Make sure to output (print out) the status of each thread before and after being interrupted. **Submission:**
1. The output from each program i.e. three parts (you may use screenshots).
2. The source code for each program in (.java) format.