**Report for Part 1 of Programming Assignment 2: Thread**

**File name in zipped folder:**  substring.c

**Problem Statement:** In this program we have to read the file that consists of two strings, namely, s1 and s2, and then find how many instances of s2 occur in s1, substrings with the use of threading library.

**Choice of threading library:** I have used with the pthread library of c to create a program. I initially picked this one due to my familiarity with this library from previous course work in CSE 1325 and other reason is efficient compatibility with the virtual machine of course including the number of processors.

**Threading in this program:**  Since it’s a multi- threaded program, we globally defined the number of threads to use at each execution of the program. String s1 is partitioned into number of threads and these threads independently search for matching string s2. Once all the threads are created and performed the calculations using num\_substring function, they add their independent total to the final total which is globally declared. This final total is guarded with mutex lock and unlock to overcome the race condition.

**Evaluation Metrics:** Time taken in execution with respect to the number of threads defined is used to evaluate the performance.

**Design of the experiment:** It’s simple, we need to change the value of the number of threads to use which is defined globally in program. Also, to record the time of execution, start and end points to calculate the execution time have also been defined using sys/time.h library in C. The number of processors set for the VM is 2.

**Collection of Data:** The program is compiled and executed ten times for each following value of number of threads 1,2,4. Then the average of ten times is taken to plot the graph presented below.

**Graph:**

**Conclusion from Graph:** From the graph it can be observed that the elapsed time reduces as no of threads are increased. Well, it cannot be a definitive pattern because it depends upon the system resources at that instance and also the processor and. Time can also vary depending upon the I/O. One important learning from this is that we can use threads to maintain and efficiently use the multiple processor for computation and I/O simultaneously, minimising the system resource usage.