Assignment 2 – Report

Name: Deepa Hegde
PSU ID: 954680127

Timing study

	1000	10000	100000	1000000	100000000
prime	0.04511	0.449875	4.31312	16.5865	Segmentation fault
Prime-omp	The other values are in the attached output files from running the shell scripts				Segmentation fault

Rest of the timing results are in the attached files along with the other code. I have used a shell script to run the programs for all the mentioned inputs and number of threads combination and have directed that output to a file.

Observations:

All the programs give correct number of prime results.

I observed segmentation fault for prime.cpp and prime-omp.cpp when the input N = 100000000.

There is no hanging of the threads in any of the programs.

I did face a lot of issue in the 4th question where the synchronization between the main thread and worker thread was needed using condition variable. Some threads who were waiting was never notified because by that time the main thread was done running. Therefore, I added a bool variable “allfound” to check if the total sieves were found. Once found then the threads need not have to wait anymore.

In the 5th question – I faced issue when the threads were not terminating. For this I used the logic which we used during assignment 1 where we add -1 and terminate. Also I have initialized a map<int,bool> to keep track of what sieves were worked on. I checked this and made sure not one than one thread worked on the same sieve.

With regards to the timing – I did not see a huge difference in the performance. There are other additional computations that we are doing to assign threads, make threads wait for the sieve. All this adds up to the elapsed time as the thread increases. A thread creation cost and context switching and additional locks on the shared variable may add an overhead to the performance. When we have a huge computational task, parallel programs certainly excels. However, when we have large data (number) to process, an ideal number of threads can improve the performance. However, it is also subjected to the hardware support

Therefore the naïve prime-par1.cpp is the best one since we are assigning partitions to threads and they work independently. There is no waiting time and no additional computation for the termination of the threads