Assignment Report

N	P	Prime.cpp	Prime-omp.cpp	Prime-par1.cpp	Prime-par2.cpp	Prime-par3.cpp	Prime-omp2.cpp	
1000	10	0.036818ms	3.87511ms	0.705412ms	0.597249ms	0.716509ms	0.03388 ms	
10000	100	0.351331ms	110.363ms	1.98664 ms	0.500824ms	1.13819 ms	8.99363 ms	
100000	1000	3.74609 ms	19.8667ms	14.0971 ms	10.2235 ms	5.28613 ms	11.87899 ms	
1000000	10	17.7238 ms	39.8409ms	64.3855 ms	49.8951 ms	33.6646 ms	45.985543ms	

From the timing report given above, it can be seen that there is a significant improvement as we increase the number of threads.

This can be accounted for the fact that as we increase the number of threads, the processing power is increased significantly.

As far as the questions are concerned, I found the prime-par2.cpp part to be extremely challenging.

My solution is unreliable as well. This is because even though sieves are being calculated properly, the total might not be properly calculated due to interference by other threads.

Also, The fact that when delay is introduced, the program seems to run properly also gives truth to my assumption that there is something wrong with one of the shared variable.

As far as other programs are concerned, they are running perfectly fine and are producing the expected results.

I also observed one thing, whenever we run a program twice with the same parameters, they seem to run faster.

This can be attributed to affinity among the cores which can lead to faster results but there is something worth noticing.

Also, in general, OpenMp runs inconsistently in terms of performance.

Sometimes it is very very fast and sometimes it is very slow. This could be due to the integration model that it follows.

Also, OpenMP generally produces results which are not reliable and is very difficult to implement as compared to the normal threaded version.