CSCI 4220 Assignment 4

MPI Prime Finder

Due Date: Monday, April 30, 11:59:59 PM

Your task for this team-based (max. 2) assignment is to implement a prime number finding program that utilizes multiple cores via MPI.

Prime numbers are useful in a variety of applications but probably most importantly in cryptography. For example, you may wish to look over the Wikipedia page for RSA. Your program should return the number of unique primes that it was able to find within the given time limit. Signal-handling code will be provided in the starter file.

As far as the method used for determining primes, there are many possible approaches. One simple approach is to try dividing your candidate n by values $2, 3, ...\sqrt{n}$. If the remainder is zero, then your n value is not prime. There are likely faster algorithms for producing prime values.

This may take a potentially long amount of time. The sizeable amounts of computation make this problem a good candidate for parallelizing via MPI. Fortunately, there is very little coordination required among the different ranks although the primes should be printed out in order. Additionally, should you exhaust all (unsigned) 32-bit primes, your program can stop. Otherwise, a signal will be sent to your program after a fixed amount of time (likely 1-2 minutes).

Please include a README.txt file which should include your name, the name of your partner, and any helpful remarks for the grader. Also please include a speedup graph as the number of ranks is increased and explain any unexpected results. Hopefully everyone can run their program on 1, 2, and 4 cores at least.

Note that due to our signal handling requirements for this assignment we will be using Open MPI. Open MPI natively supports signals whereas some other implementations do not pass them through to your binary.

Please make sure to get started early!

So we don't anger the Submitty gods, instead of saving gigabytes of integers, we can instead use the number of primes less than a specific value as the following output demonstrates: Sample output given below:

```
bash$ mpirun-openmpi-mp -np 2 ./a.out
                N               Primes
               10                    4
              100                   25
             1000                  168
<Signal received>
             1020                  171
```