Documentation

Problem Statement (30 points)
Your non-technical manager assigns you the task to find all primes between 1 and 10^8.The assumption is that your company is going to use a parallel machine that supports eight concurrent threads. Thus, in your design you should plan to spawn 8 threads that will perform the necessary computation. Your boss does not have a strong technical background, but she is a reasonable person. Therefore, she expects to see that the work is distributed such that the computational execution time is approximately equivalent among the threads. Finally, you need to provide a brief summary of your approach and an informal statement reasoning about the correctness and efficiency of your design. Provide a summary of the experimental evaluation of your approach. Remember, that your company cannot afford a supercomputer and rents a machine by the minute, so the longer your program takes, the more it costs. Feel free to use any programming language of your choice that supports multi-threading as long as you provide a ReadMe file with instructions for your manager explaining how to compile and run your program from the command prompt.

IMPLEMENTATION:

- Initialized MAX to 10^8 and set counter to MAX-1.
- Skipped all the even numbers to improve efficiency of the code. All loops are incremented or decremented by a factor of 2 to skip even numbers.
- The top 10 prime numbers are found using the while loop without any parallel execution with a count of 10 and saved in array till count reaches 0.
- For the rest of the number the counter is decremented till 3 and number of primes and the sum of primes is incremented accordingly in critical sections to avoid incorrect data or race conditions.
- Number of threads is set to 8 according to the problem in the #pragma for loop and can be changed according to the requirement.
- As there are many multiples of 3, 5, 7, ... the smaller prime numbers their multiples are skipped till 13 for increasing efficiency.
- The #pragma omp critical makes sure only one tread can modify it at any given time.
- In the end since all evens are skipped 2 is added and then the final sum and total primes is printed.

PROOF OF CORRECTNESS:

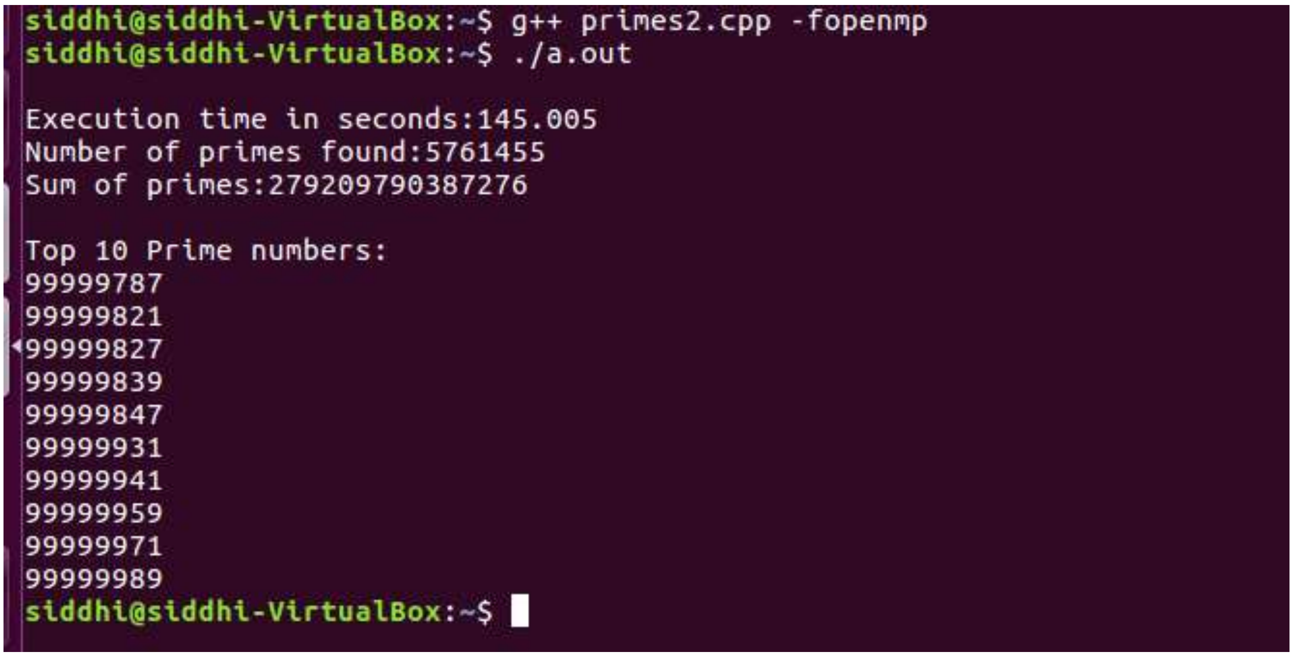
Same results when the code is run sequentially with num_threads changed to 1.

RESULTS:

Configuration of system used to execute:

Intel Dual Core i7-7500U CPU @ 2.70GHz
RAM: 16 GB

Parallel Execution Time: ~ 145 seconds.

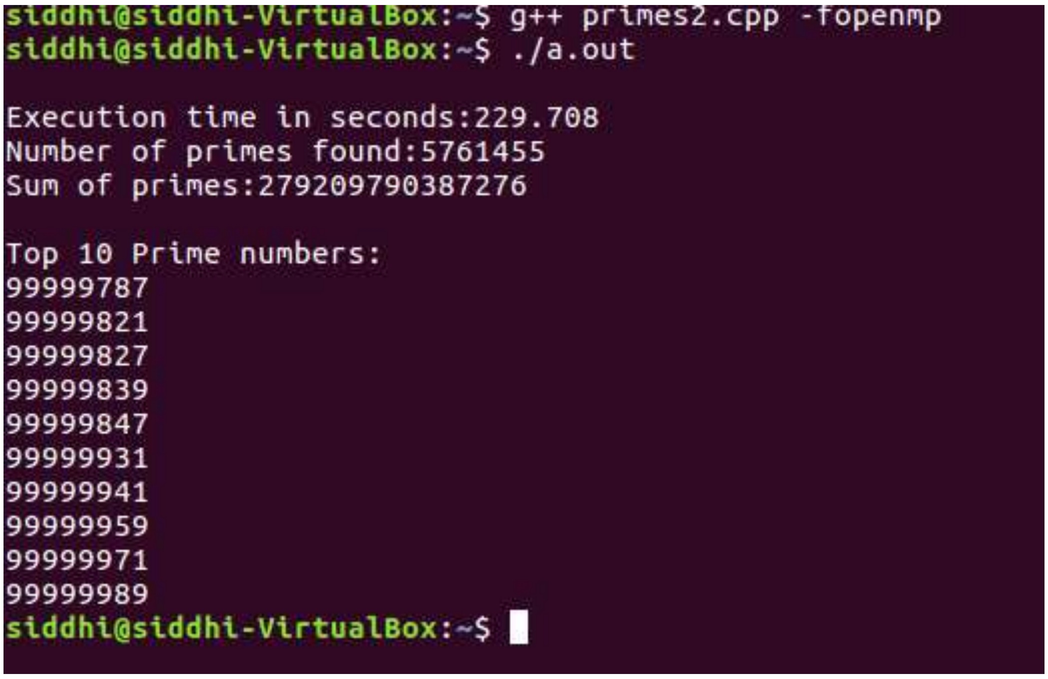


```
siddhi@siddhi-VirtualBox:~$ g++ primes2.cpp -fopenmp
siddhi@siddhi-VirtualBox:~$ ./a.out

Execution time in seconds:145.005
Number of primes found:5761455
Sum of primes:279209790387276

Top 10 Prime numbers:
99999787
99999821
99999827
99999839
99999847
99999931
99999941
99999959
99999971
99999989
siddhi@siddhi-VirtualBox:~$
```

Sequential Execution Time: ~ 230 seconds.(with number of threads = 1)



```
siddhi@siddhi-VirtualBox:~$ g++ primes2.cpp -fopenmp
siddhi@siddhi-VirtualBox:~$ ./a.out

Execution time in seconds:229.708
Number of primes found:5761455
Sum of primes:279209790387276

Top 10 Prime numbers:
99999787
99999821
99999827
99999839
99999847
99999931
99999941
99999959
99999971
99999989
siddhi@siddhi-VirtualBox:~$
```

The times returned are "per-thread times" by which is meant they are not required to be globally consistent across all the threads participating in an application.