Graphics Processing Units (GPUs): Architecture and Programming
Programming Assignment 3

In this lab you will write CUDA code to generate the prime numbers factors of a non-zero integer N.

General notes:

- The name of the source code file is: factorize.cu
- Write your program in such a way that to execute it I type: *./factorize N*
  Where N is a positive number bigger than 1 and less than or equal to maximum unsigned integer (2^{32}-1).
- The output of your program is printed on the screen as: a b c … where a *b*c*… = N and where all a, b, c, … are prime numbers. Note that a=b is possible. For example, if N = 4 then the output is 2 2.
- You can assume that we will not do any tricks with the input.
- If N itself is prime, you can just print N.

The sequential algorithm for generating prime numbers:

```
Input: N
1) While N is divisible by 2, print 2 and divide N by 2.
2) After step 1, N must be odd. Now start a loop from i = 3 to ceiling(square root of N).
   While i divides N:
     • print i
     • keep dividing N by i and print i till N is no longer divisible by i
     • increment i by 2
     • continue the outer loop
3) If the remaining of N is a prime number and is greater than 2, then N will not become 1 by above two steps. So print N if it is greater than 2.
```

Now, you need to think a bit to find a way to do the same in parallel. This is not as easy as it seems! Feel free to optimize the above algorithm in any way you want.

How to measure the performance and scalability of your code?

To see how efficient your implementation is, you need to compare against a sequential version. Therefore, do the following:

1. Implement a C version called seqfactorize.c that takes the same argument as the CUDA version. That is: ./seqfactorize N
2. Implement the CUDA version.
3. Run both program for the same input and measure the overall time using the *time* command in Linux. That is: *time ./factorize N*
   Time will return 3 numbers: usr, system, and real. You will need to report the real.
4. Try with different values of N up to maximum integer and see how the GPU version and CPU version compare. You may need to repeat the same experiment 5 times and take the median to get a more precise result.

The report
Write a report that contains the following:

- [10 points] A list of all the tricks (CUDA and non-CUDA) you have used to speedup the parallel version.
- [5 points] For the GPU version: What is done on CPU? What is done on GPU?
- [20 points] A bar with N values as x-axis and showing the speedup (y-axis) (CPU time / GPU time).
  Try with N with the following values:
  10
  100
  1,000
  10,000
  100,000
  1000,000
  10,000,000
  100,000,000
  1000,000,000
  2000,000,000
- [10 points] In simple bullet points, explain the results in the above graph. Do not say the CPU version is better than GPU version for small N because of the communication overhead. But show numbers on the communication overhead and how it increases with N. Are there any branch divergence? How is global memory access in gpu affecting performance? … etc. Feel free to use any profiling tool you want (e.g. Nsight) .Correct analysis always means numbers not words!

What do you have to submit:
A single zip file. The file name is your lastname.firstname.zip
Inside that zip file you need to have:

- seqfactorize.c [10 points]
- factorize.cu [45 points]
- pdf file containing the graphs and explanation. [45 points]

Submit the zip file through Brightspace.

Enjoy!