Homework 4
Due July 30, 2019

Please solve the following and upload your solutions to your private GitHub repository for the class as homework4.pdf by 11:59pm on the due date above. If for some reason this poses a technical problem, or you wish to include diagrams that you don’t wish to spend time drawing in a drawing application, you may hand in a printed copy (not hand-written) at the beginning of class (5:45pm) on the day of the deadline. **Unlike labs, late homeworks will be assigned a grade of 0.**

This homework is intended to assist in your studying for the Midterm Quiz, so I recommend trying to complete it before the Quiz. Remember, you **must** work on homeworks alone: if you confer with any other students on the homework, either to ask questions or to work together, you **must** cite those students in your solutions. You **must** also use your own words, and any other sources (internet, etc.) beyond the class lectures **must** be cited.

1. **Performance:** Consider the performance of a program responsible for taking a set of 10,000 large numbers (e.g., between $1e20$ and $1e40$), and determining the prime factors of each number. A user has a four-core machine, and wants to run this program in the background while they play a fast-paced game (that spawns two threads, one for the game logic and one for rendering) at the same time.
   a. What kind of bounds (minimum and maximum) can you give on the number of threads the prime factor program should spawn? Why not fewer or more?
   b. What performance metric(s) would you as a programmer use (e.g., latency, throughput, something else?) What would you optimize for in this situation, not just in your own program, but system-wide?

2. **Lock-Free Data Structures:** What causes the ABA problem? How can you solve it, and why does that solve it? What extra hardware and software primitive(s), if any, are required to implement a fix to the ABA problem?

3. **Thread-Safe Data Structures:** A **circular buffer** is a fixed-size data structure, often implemented with a vector, with a read pointer and a write pointer. It acts as if the end and the beginning are connected together: if you read or write past the end, you simply loop back around to the beginning. For each item written, you write to the element under the write pointer, then move that pointer
forward; for each item read, you read the element under the read pointer, then move that pointer forward. The number of items currently in the circular buffer is the distance between the read and write pointers: if they’re at the same element, there’s nothing to read, if the write pointer is one element beyond the read pointer, there’s one element to read, etc.

Imagine that a circular buffer has three methods:

- **push()**: if the buffer is not full, write to the element under the write pointer, then move the write pointer forward.
- **pop()**: if the buffer is not empty, read from the element under the read pointer, then move the read pointer forward.
- **size()**: return the number of items in the buffer (not the capacity of the underlying vector).

Without actually implementing a thread-safe circular buffer, discuss:

a. **Warmup**: in as few lines as possible (e.g., one line for each of the three methods above), how could you make this thread-safe using C++ mutexes? Just say what you would add! Don’t actually write the full methods!

b. Could you make this more performant with one or multiple read-writer lock(s)? If so, how? If not, why not?

c. Could you make this more performant with CAS? If so, how? If not, why not. **Outline your technique. Do not actually solve the full implementation.**

4. **Correctness**: Consider the following two threads accessing shared global variable `thr_glob`. (a) What kind(s) of concurrency bug(s) are present in this code? (b) **Describe** (ie, don’t just list) two possible ways to fix this. If it helps, use code to demonstrate why these are good fixes.

<table>
<thead>
<tr>
<th>Thread 1</th>
<th>Thread 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>if (thr_glob-&gt;proc_info) {</td>
<td>thr_glob-&gt;proc_info = nullptr;</td>
</tr>
<tr>
<td>fputs(thr_glob-&gt;proc_info);</td>
<td>}</td>
</tr>
</tbody>
</table>