1. Warm up: Running on some specific machine $X$ with set hardware and software, how many threads should your program spawn, given that it has enough parallel tasks to run to occupy many threads? Think about how the hardware (eg, memory and number of cores) and software (ie, programs already occupying some resources) affect the number of threads you should spawn. Don’t overthink this one!

2. Explain (in your own words!) how Peterson’s Algorithm works. Why does it need to be re-written for more than 2 threads?

3. If multiple on-die caches (eg, L2 and L3 cache) are higher-latency than an L1 cache, why do we have them anyway? (Hint: how big and slow is main memory? How big and slow is the L1 cache?)

4. Consider the following code:

```c
1 static int sum_stat_a = 0;
2 static int sum_stat_b = 0;
3 int aggregateStats(int stat_a, int stat_b) {
4     sum_stat_a += stat_a;
5     sum_stat_b += stat_b;
6     return sum_stat_a + sum_stat_b;
7 }    
8 void init(void) { }
```

Use a single pthread mutex to make this function thread-safe. Add global variables and content to the `init()` function as necessary.

5. Let’s make this more parallelizable. We always want to reduce critical sections as much as possible to minimize the time threads need to wait for a resource protected by a lock. Modify the original code from question 3 to make it thread-safe, but use two mutices this time, one for `sum_stat_a` and one for `sum_stat_b`.

6. What’s the difference between `join()`ing and `detach()`ing a pthread? When would we want to use each technique?