Lecture: Replication and Consistency
Exercise Sheet 1
https://pl.cs.uni-kl.de/homepage/de/teaching/ws19/rac/

1 Amdahl's Law

You have a choice between buying one uni-processor that executes five zillion instructions per second, or a 8-processor multiprocessor where each processor executes one zillion instructions per second. Using Amdahl's Law, explain how you would decide which to buy for a particular application.

2 Mutual Exclusion

Programmers at the Flaky Computer Corporation designed the following protocol to achieve n-thread mutual exclusion.

Flaky Lock
Initially: `int turn = 0; boolean busy = false;`
Program for Process `i`:

```
do
  do
    turn = i;
  while (busy);
  busy = true;
while (turn != i);

//critical section

busy = false;
```

For each question, either sketch a proof, or give an execution where it fails.

- Does this protocol satisfy mutual exclusion?
- Is this protocol starvation-free?
- Is this protocol deadlock-free?

3 Uncontended Locks

In practice, almost all lock acquisitions are uncontended, so the most practical measure of a lock's performance is the number of steps needed for a thread to acquire a lock when no other thread is concurrently trying to acquire the lock.
Scientists at TU Kunterbunt have devised the following wrapper for an arbitrary lock `L`:

Fast-Path Lock
Initially: `int x, y = -1;`
Program for Process `i`:

```
x = i;                // I'm here
wait (y != -1) {}     // is the lock free?
  y = i;               // me again?
  if (x != i)          // Am I still here?
    L.lock();      // slow path
}

//critical section

y = -1;
L.unlock();
```

They claim that if the base Lock `L` provides mutual exclusion and is starvation-free, so does the `FastPath` lock, but it can be acquired in a constant number of steps in the absence of contention. Sketch an argument why they are right, or give a counterexample.

4 Tree locks

A way to generalize the two-thread Peterson lock to n threads is to arrange a number of two-thread Peterson locks in a binary tree.
For simplicity, suppose that $n = 2^k$ for some k. Each thread is assigned a leaf lock which it shares with one other thread. Each lock treats one thread as thread 0 and the other as thread 1.
In the tree-lock's acquire method, the thread acquires every two-thread Peterson lock from that thread's leaf to the root. The tree-lock's release method unlocks each fo the two-thread Peterson locks which the thread has acquired, from the root back to its leaf. At any time, a thread can be delayed for a finite duration of time. (This means, threads can take naps, or even vacations, but they do not drop dead.)
For each property, either sketch a proof why it holds, or describe a (possibly infinite) execution where it is violated:

- mutual exclusion,
- deadlock-freedom, and
- starvation-freedom.

Is there an upper bound on the number of times the tree-lock can be acquired and released between the time a thread starts acquiring the tree-lock and when it succeeds?