CMU 15-418/618: Parallel Computer Architecture and Programming
Practice Exercise 4

Problem 1: Scheduling Requests

A. (5 pts) You and your business partner buy a **single-core computer** to host your startup's web site. Your site receives exactly one type of request: it involves a disk read with latency 900 ms (during which the processor is idle, and waits) followed by 100 ms of processing. Thus, its response latency is 1 sec.

As your business grows you start receiving requests at steady rate of one request every 100 ms. Your partner claims, "Let's buy nine more machines so we can keep up!" You claim, "Wait a minute, we can save money and handle this load without all those machines!" Who is right? How many machines do you actually need to service 10 requests per second, and how would you architect your web server to achieve this throughput? **Assume that disk I/O bandwidth is infinite, multiple I/O operations can be outstanding at once, and that all requests are unique and so optimizations like caching are not possible.**

B. (2 pts) Immediately after announcing THE DEAL, Prof. Kayvon realizes he has a throughput problem. Students are arriving to take the deal at a **rate of one per minute**, but Kayvon can only administer deals at a **rate of one per ten minutes**. The line is growing!!! Assume Kayvon can be cloned at a cost of $1M per clone. Assuming multiple Prof. Kayvons can process different students perfectly in parallel, how much does it cost to keep the line from growing as more students arrive? (Clarification: Assume that your system consists of the original Prof. Kayvon plus any required clones.)

C. (3 pts) At 8:30am, you go to grab a coffee at the Gates 3rd floor cafe. There is no line, so you immediately walk up and order. Later that day, you go to get your daily double-creme, super-duper, organic soy latte between classes at 2:50pm to take to 418/618, and the line is 15 students long. You tell the manager, "you really need to hire more workers,". She responds, "I can't, because it would be cost too much to double my staff for the 8-5pm workday". Using the words "throughput" and "elasticity" in your answer, make a suggestion on how the manager might run her business.

Problem 2: Load Linked / Store Conditional

A common set of instructions that enable atomic execution is load linked-store conditional (LL-SC). The idea is that when a processor loads from an address using a `load_linked` operation, the corresponding `store_conditional` to that address will succeed only if no other writes to that address from another processor have intervened. Note that unlike `test_and_set` or `compare_and_swap`, which are single atomic operations, load linked and store conditional are different operations and the processor may execute other instructions in between these two operations. Pseudocode for these instructions is given below.

```
int load_linked(int* addr) {
  return *addr;
}

bool store_conditional(int* addr, int new_val) {
  if (data in addr has not been changed since the corresponding load_linked) {
    *addr = new_val;
    return true;
  } else
     return false;
}

Example usage:

int x;

load_linked(&x);
y = f(x);  // do stuff with x here
store_conditional(&x, y);
```

A. (5 pts) Implement a spin lock using LL and SC primitives:

```
void Lock(int* l) {









}

void Unlock(int* l) {



}
```

B. (5 pts) Given that you now have a good understanding of a basic implementation of invalidation-based cache coherence, describe how you would extend the behavior of MESI caches to implement the load-linked and store conditional instructions.