[6 points] When two or more threads are blocked on a call to **mutex::lock**, any one of them might be selected to acquire the lock once the **mutex** becomes available. Restated, the **mutex** isn't obligated to maintain any sort of FIFO queue to ensure the thread waiting longer than any other is chosen first.

A **strong** mutex, or a **smutex**, ensures that blocked threads are woken up in the same order they are blocked. There are many **smutex** implementations, and one that relies on a queue of **condition_variable_any**s is presented below (interface on the left, implementation on the right).

```
// smutex.h
class smutex {
  public:
   void lock();
   void unlock();

  private:
    mutex m;
    list<condition_variable_any *> queue;
};
```

```
// smutex.cc
void smutex::lock() {
  condition_variable_any cv;
  unique_lock<mutex> ul(m);
  queue.push_back(&cv);
  while (queue.front() != &cv) {
    cv.wait(m);
  }
  queue.pop_front();
}

void smutex::unlock() {
  unique_lock<mutex> ul(m);
  if (!queue.empty()) {
    queue.front()->notify_all();
  }
}
```

Study the implementation of the **smutex** methods and answer the following questions:

- [2 points] Does the implementation guarantee that a thread calling **smutex::lock** before any others gets the lock on the **smutex** first? Why or why not?

  *Technically not, since the same ordering non-guarantee that comes up with traditional mutexes is present in **unique_lock<mutex> ul(m)**. (That's all they need to say.). However, as opposed to arbitrary mutexes, **smutex::m** is locked down for a very, very, very short window, so the probability of FIFO happening is much, much higher. (If they say this, then they get credit as well.).*

- [2 points] In **unlock** when the code calls **queue.front()->notify_all(),** could we instead notify just one waiting thread at that point instead of all of them without impacting functionality? Very briefly justify your answer.

  Yes, because exactly one thread is waiting on the notified condition variable, so notifying one thread would notify everyone who is waiting.

- [2 points] Can the **`queue.pop_front()`** line in **`smutex::lock()`** be moved so that it's the last line in **`smutex::unlock()`** instead? Why or why not?

  *No - if the leading address is popped before the blocked threads wakes up and evaluates **`queue.front() == &cv`**, the test will fail, the blocked thread will forever deadlock.*