第三次作业

学号	姓名
20319045	刘冠麟

q1

1. **线程将发生死锁。**

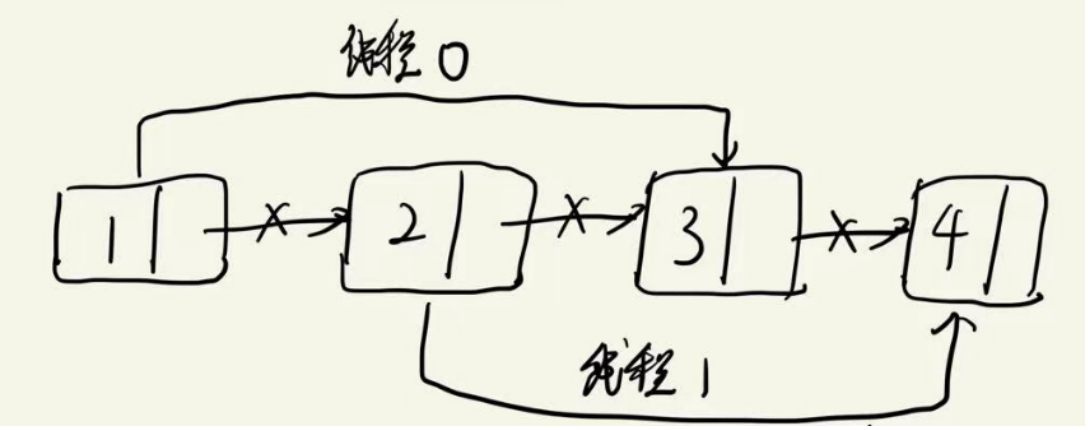
   这个时候线程0将等待获取互斥量mut1，而线程1将等待获取mut0，但由于线程0正在等待mut1，线程0无法释放mut0，而此时线程1又在正在等待获取mut0，也无法释放mut1。

2. **还会有问题。**这种情况下每个线程都会等待另一个线程去改变一个标志变量。

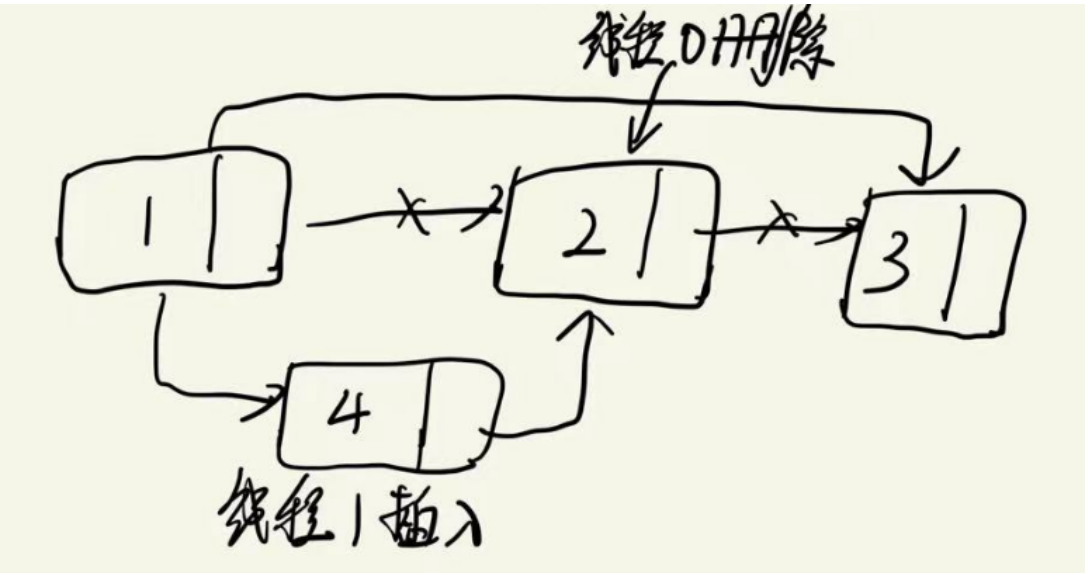
3. **有。**因为这种情况下两个线程都会被阻塞在sem_wait调用中，因此没有线程能调用sem_post。

q2

1. 两个线程同时删除会引发问题，如下图：

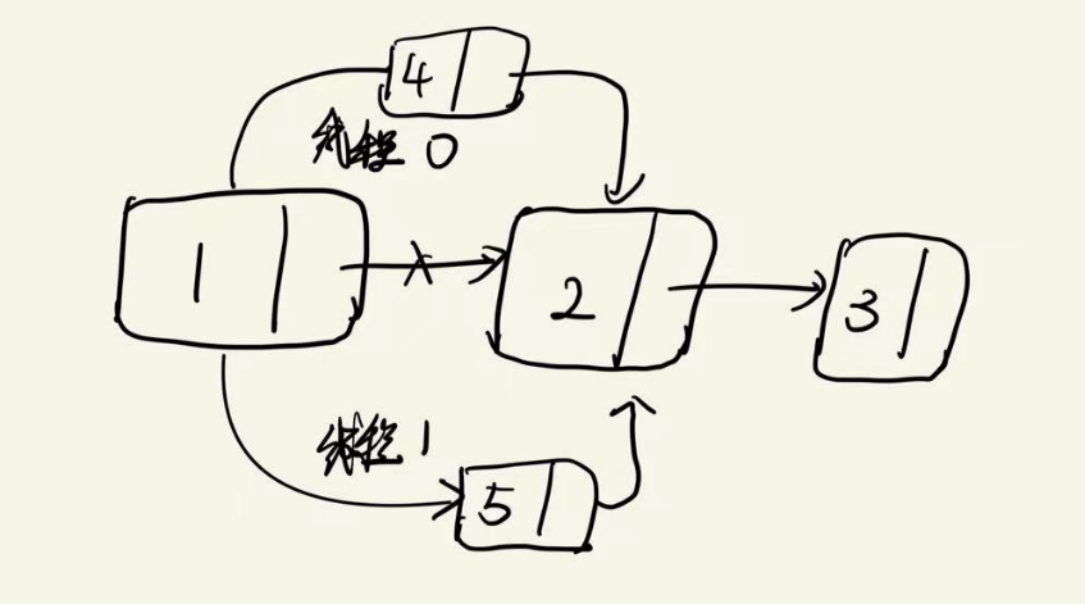


2. 可能有问题，如下图：



3. 会有问题，比如线程0在访问一个节点时，这个节点正在被线程1删除。当线程0在线程1完成删除后试图移动到下一个节点时线程0解引用的指针可能会引起段错误。

4. 这可能会有问题，如下图：



5. 会有问题。当线程1正试图插入一个新节点时，如果线程0正在进行Member操作，如果线程1在线程0确定列表中没有该值到线程0返回之间的时间内插入了被搜索的值，线程0可能会错误地返回false。

q3

在第一阶段中使用一个读锁来锁链表是安全的，在这个阶段只需要读取链表数据而不进行任何修改，只要没有线程试图修改链表，多个线程同时读取不会造成数据不一致的问题。

但是尽管第一阶段使用读锁是安全的，从读锁到写锁的转换时也**有可能导致链表状态不一致。**如果一个线程在第一阶段结束后需要进入第二阶段进行修改，它必须先释放读锁然后获取写锁，在转换过程中链表的状态可能被其他线程改变（比如线程在等待写锁时其他线程可能插入或删除节点）

q4

缓存行大小为 64 字节，每个 double 类型数据占用 8 字节。因此，每个缓存行可以存储 64 / 8 = 8 个 double 类型数据。

根据题目有四个线程，线程具体分配如下

- **线程 0** 处理 y 的元素从 y[0] 到 y[1999]。
- **线程 1** 处理 y 的元素从 y[2000] 到 y[3999]。
- **线程 2** 处理 y 的元素从 y[4000] 到 y[5999]。
- **线程 3** 处理 y 的元素从 y[6000] 到 y[7999]。

线程 0 和 线程 2

线程0 处理的元素范围是 y[0] 到 y[1999]，线程2处理的元素范围是 y[4000]到 y[5999]，线程 0 的最后一个元素是 y[1999]，而线程 2 的第一个元素是 y[4000]。

给出缓存行可以包含元素，线程0处理的最后一个元素 y[1999] 和线程2处理的第一个元素 y[4000]在索引上相差2001，远大于一个缓存行所能包含的元素数（8个double）。因此，**线程0和线程2之间不会发生伪共享。**

线程 0 和 线程 2

线程 3 的第一个元素是 y[6000]。同样根据元素索引，线程 0 的缓存行不可能与线程 3 的缓存行重叠。**线程 0 和线程 3 之间也不会发生伪共享。**

q5

1. 在一个系统中有两个双核处理器，且同一个处理器上的所有核共享一个缓存，可以选择两个线程分配给一个处理器。考虑到线程间的对称性（例如，选择线程 2 和 3 分配给一个处理器等同于选择线程 0 和 1），总共有**三种不同的方式**分配线程到处理器：
    - 线程 0 和 1 在一个处理器上，线程 2 和 3 在另一个处理器上。
    - 线程 0 和 2 在一个处理器上，线程 1 和 3 在另一个处理器上。
    - 线程 0 和 3 在一个处理器上，线程 1 和 2 在另一个处理器上。

    这些分配方式确保了每个处理器上都有两个线程运行。
2. **有。**考虑到每个缓存行可以存储 8 个 double 类型数据，伪共享发生在当一个缓存行中的数据被一个处理器上的线程写入时，影响到另一个处理器上线程读取的同一缓存行的数据。

    如果按照这种情况分配： y[0] 到 y[7] 分配在两个不同的缓存行中，且不同的处理器上的线程操作的数据分别属于这两个缓存行。例如：
    - 缓存行 A 包含 y[0] 到 y[7]，线程 0 和 1 操作 y[0] 到 y[3]。
    - 缓存行 B 包含 y[4] 到 y[7]，线程 2 和 3 操作 y[4] 到 y[7]。

    如果这样的分配实现，每个处理器上的线程只会操作它们共享的缓存行内的数据，因此不会影响另一个处理器上缓存行的数据。这种分配避免了伪共享，因为没有线程需要访问另一个处理器上线程正在写入的缓存行。