HW4

> 读者写者问题（写者优先）：
>
> 1）共享读
>
> 2）互斥写、读写互斥
>
> 3）写者优先于读者（一旦有写者，则后续读者必须等待，唤醒时优先考虑写者）

```
int readerCount = 0;         // 读者数量
int writerCount = 0;         // 写者数量
semaphore readerMutex = 1;   // 信号量，用于readerCount的互斥
semaphore writerMutex = 1;   // 信号量，用于writerCount的互斥
semaphore dataMutex = 1;     // 信号量，用于data的互斥
semaphore noWriter = 1;      // 信号量，用于在没有writer的时候唤醒reader

// 写者
function writer() {
    P(writerMutex);
    writerCount++;
    if (writerCount == 1) {
        P(dataMutex);
    }
    V(writerMutex);

    writing...

    P(writerMutex);
    writerCount--;
    if (writerCount == 0) {
        V(dataMutex);
        V(noWriter); // 没有writer，唤醒reader
    }
    V(writerMutex);
}

// 读者
function reader() {
    P(writerMutex);
    if (writerCount != 0) { // writer优先
        V(writerMutex);
        P(noWriter);
    } else {
        V(writerMutex);
    }
    P(readerMutex);
    readerCount++;
    if (readerCount == 1) {
        P(dataMutex);
    }
    V(readerMutex);

    reading...
```

```
    P(readerMutex);
    readcount--;
    if (readcount == 0) {
        V(dataMutex);
    }
    V(readerMutex);
}
```

> 寿司店问题。假设一个寿司店有 5 个座位，如果你到达的时候有一个空座位，你可以立 刻就坐。但是如果你到达的时候 5 个座位都是满的有人已经就坐，这就意味着这些人都是一起来吃饭的，那么你需要等待所有的人一起离开才能就坐。编写同步原语，实现这个场景的 约束。

```
int eating = 0;
int waiting = 0;
semaphore mutex = 1;
semaphore waitline = 0;
bool mustWait = false;

function customer() {
    P(mutex);
    if (mustWait) { // 需要等位
        waiting += 1;
        V(mutex);
        P(waitline);
    } else { // 直接吃
        eating += 1;
        mustWait = eating == 5;
        V(mutex);
    }

    eating...

    P(mutex);
    eating -= 1;
    if (eating == 0) {
        call = min(5, waitline);
        waitline -= call;
        eating += call;
        mustWait = eating == 5;
        for i in 1...n { // 唤醒n个人
            V(waitline);
        }
    }
    V(mutex);
}
```

> 进门问题。（1）请给出 P、V 操作和信号量的物理意义。（2）一个软件公司有 5 名员工， 每人刷卡上班。员工刷卡后需要等待，直到所有员工都刷卡后才能进入公司。为了避免拥挤， 公司要求员工一个一个通过大门。所有员工都进入后，最后进入的员工负责关门。请用 P、 V 操作实现员工之间的同步关系。

1. P（Wait）操作的物理意义是尝试获取资源。当一个线程执行P操作时，它试图将信号量的值减少1。如果信号量的值大于0，则可以获取资源并继续执行；如果信号量的值为0，则线程将被阻塞，直到资源可用为止。

2. V（Signal）操作的物理意义是释放资源。当一个线程执行V操作时，它将信号量的值增加1，表示释放一个资源。这样，其他线程如果在等待这个资源，就可以继续执行了。

```
// 初始化信号量
semaphore door = 0;
semaphore countMutex = 1;
int count = 0;

// 员工线程
function employee(int id) {
    P(countMutex); // 刷卡
    count++;
    if (count != 5) { // 不是最后一个
        V(countMutex);
        P(door); // 等着进门
        enter... // 进入
    } else {
        V(countMutex);
        for i in 1..4 {
            V(door); // 放进去前四个人
        }
        enter... // 进入
        closeDoor... // 关门
    }
}
```

> 搜索-插入-删除问题。三个线程对一个单链表进行并发的访问，分别进行搜索、插入和删 除。搜索线程仅仅读取链表，因此多个搜索线程可以并发。插入线程把数据项插入到链表最 后的位置；多个插入线程必须互斥防止同时执行插入操作。但是，一个插入线程可以和多个 搜索线程并发执行。最后，删除线程可以从链表中任何一个位置删除数据。一次只能有一个 删除线程执行；删除线程之间，删除线程和搜索线程，删除线程和插入线程都不能同时执行。 请编写三类线程的同步互斥代码，描述这种三路的分类互斥问题

```
insertMutex = Semaphore(1)
noSearcher = Semaphore(1)
noInserter = Semaphore(1)
searchSwitch = Lightswitch()
insertSwitch = Lightswitch()

搜索者:
searchSwitch.wait(noSearcher)//对noSearcher上锁
# critical section //多个searcher互斥
searchSwitch.signal(noSearcher)

插入者:
insertSwitch.wait(noInserter)
insertMutex.wait() //多个inserter要互斥访问
# critical section
insertMutex.signal()
insertSwitch.signal(noInserter)

删除者:
noSearcher.wait()
noInserter.wait()
```

```
# critical section
noInserter.signal()
noSearcher.signal()
```