

## Q1

```
1  int count = 0;
2  semaphore mutex = 1;
3  semaphore rw = 1;
4  semaphore w = 1;
5
6  writer() {
7      while(1) {
8          P(w);
9          P(rw);
10         writing;
11         V(rw);
12         V(w);
13     }
14 }
15
16 reader() {
17     while(1) {
18         P(w);
19         P(mutex);
20         if (count==0) {
21             P(rw);
22         }
23         count++;
24         V(mutex);
25         V(w);
26         reading;
27         P(mutex);
28         count--;
29         if (count==0) {
30             V(rw);
31         }
32         V(mutex);
```

```

33     }
34 }

```

## Q2

```

1  int count = 0;
2  semaphore number = 6;
3  semaphore waiting = 0;
4  semaphore mutex = 1;
5
6  customer() {
7      P(number);
8      P(mutex);
9      if(count==5) {
10         p(waiting);
11     }
12     count++;
13     V(mutex);
14     eating;
15     P(mutex);
16     count--;
17     if(count == 0) {
18         V(waiting);
19     }
20     V(mutex);
21     V(number);
22 }

```

## Q3

(1): 要证明  $((A \rightarrow B) \rightarrow A) \rightarrow A$ , 我们可以使用题目中给出的定理和一些额外的逻辑定律: 肯定后件律、蕴含词分配律、换位律。

1.  $A \rightarrow A$  (自反律) 2.  $(P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P)$  (换位律) 3.  $Q \rightarrow (P \rightarrow R), P \vdash Q \rightarrow R$  (假言三段论) 4.  $P \rightarrow Q, Q \rightarrow R \vdash P \rightarrow R$  (假言推理) 5.  $A \rightarrow B, A \vdash B$  (肯定后件律) 6.  $(A \rightarrow B) \rightarrow ((A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow C))$  (蕴含词分配律)

1. **\*\*目标\*\***: 证明  $((A \rightarrow B) \rightarrow A) \rightarrow A$
2. 设  $P = (A \rightarrow B) \rightarrow A$  和  $Q = A$
3. 使用假言三段论 (定理3):

$$Q \rightarrow (P \rightarrow Q), P \vdash Q \rightarrow R$$

4. 具体应用: 1.  $A \rightarrow (P \rightarrow A)$  (因为  $Q = A$  和  $R = A$ ) 2.  $P = (A \rightarrow B) \rightarrow A$
5. 根据蕴含词分配律 (定理6), 我们有:

$$(A \rightarrow B) \rightarrow ((A \rightarrow (B \rightarrow A)) \rightarrow (A \rightarrow A))$$

6. 根据换位律 (定理2), 我们有:

$$(A \rightarrow B) \rightarrow A \Rightarrow \neg A \rightarrow \neg((A \rightarrow B) \rightarrow A)$$

7. 根据肯定后件律 (定理5), 我们有:

$$A \rightarrow A$$

8. 最后, 将这些结合在一起:

$$((A \rightarrow B) \rightarrow A) \rightarrow A$$

以上步骤使用了公理3两次, 并结合其他定理完成了证明。这样我们就完成了这个命题逻辑的证明。信号量 (Semaphore) 是一个整数变量, 可理解为一个计数器, 用于控制多个线程或进程对共享资源的访问。根据信号量的取值不同, 可以有两种类型: 二进制信号量 (Binary Semaphore): 取值只能为0或1, 通常用作互斥锁 (Mutex)。计数信号量 (Counting Semaphore): 取值可以是一个非负整数, 表示可用的资源数量。

**P操作** (Proberen, 荷兰语动词“试图”的含义): P操作对应于请求或等待一个资源。当线程或进程想要获取资源时, 它会执行P操作。如果信号量的值大于0, 表示有资源可用。执行P操作会将信号量的值减去1, 然后该线程或进程会持续其执行。如果信号量的值为0, 则没有可用资源。执行P操作的线程或进程将被阻塞, 直到信号量的值再次变得大于0 (即有其他线程或进程释放资源)。

**V操作** (Verhogen, 荷兰语动词“增加”的含义): V操作对应于释放一个资源。当线程或进程完成对资源的使用后, 它会执行V操作。V操作将信

号量的值加1，表示一个资源单元变为可用状态。如果有其他线程或进程正在P操作中被阻塞，增加了信号量的值可能导致等待的线程或进程被唤醒，以便能够继续执行并访问资源。

物理意义的解释：可以把信号量想象为一个有限容量的停车场，P操作像是一辆车进入停车场，如果有空余车位（信号量大于0），车辆就进入并占用一个车位（信号量减1）。如果没有空车位（信号量为0），车辆就在入口等待。而V操作就像是一辆车离开停车场，释放了一个车位（信号量加1），如果有车辆在等待，它们就可以占用这个刚释放的车位了。

(2):

```
1  int count = 0;
2  semaphore mutex = 1;
3  semaphore waiting = 0;
4  semaphore in = 1;
5
6  employer() {
7      P(mutex);
8      count++;
9      if(count==5) {
10         V(mutex);
11         for(int i = 1;i <= 4;i++)
12             V(waiting);
13     }
14     else {
15         V(mutex);
16         P(waiting);
17     }
18     P(in);
19     if(count == 1)
20         closeDoor();
21     count--;
22     V(in);
23 }
```

Q4

```

1  int count = 0;
2  semaphore insert = 1;
3  semaphore delete = 1;
4  semaphore mutexWrite = 1;
5
6  reader() {
7      P(mutexWrite);
8      if(count==0) {
9          P(delete);
10     }
11     count++;
12     V(mutexWrite);
13     reading;
14     P(mutexWrite);
15     count--;
16     if(count==0) {
17         V(delete);
18     }
19     V(mutexWrite);
20 }
21
22 inserter () {
23     P(insert );
24     P(mutexWrite);
25     if(count==0) {
26         P(delte);
27     }
28     count++;
29     V(mutexWrite);
30     inserting ;
31     P(mutexWrite);
32     count--;
33     if(count==0) {

```

```
34         V(delete);
35     }
36     V(mutexWrite);
37     V(insert);
38 }
39
40 deleter () {
41     P(delete);
42     deleting;
43     V(delete);
44 }
```