$Q1$

```
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);
```

```
        }
}
```

Q2

```
int  count = 0;
semaphore number = 6;
semaphore waiting = 0;
semaphore mutex = 1;

customer() {
        P(number);
        P(mutex);
        if(count==5) {
                p(waiting);
        }
        count++;
        V(mutex);
        eating;
        P(mutex);
        count--;
        if(count == 0) {
                V(waiting);
        }
        V(mutex);
        V(number);
}
```

Q3

（1）：

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  exployer() {
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);
```

```
        }
        P(in);
        if (count == 1)
                closeDoor();
        count--;
        V(in);
}
```

Q4

```
int  count = 0;
semaphore insert = 1;
semaphore delete = 1;
semaphore mutexWrite = 1;

reader() {
        P(mutexWrite);
        if (count==0) {
                P(delete);
        }
        count++;
        V(mutexWrite);
        reading;
        P(mutexWrite);
        count--;
        if (count==0) {
                V(delete);
        }
        V(mutexWrite);
}

inserter () {
        P(insert );
        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      }
```