1.

```
semaphore mutex = 1;            //互斥信号量，保护缓冲区
semaphore odd = 0;          //共享缓冲区奇数
semaphore even = 0;         //共享缓冲区偶数
semaphore empty = N;        //共享缓冲区空位

void P1() {
 while(true) {
    int x = produce();
    P(empty);
    P(mutex);
    put(x);
    V(mutex);
    if (x % 2 == 0)
       V(even);
    else
       V(odd);
 }
}

void P2() {
 while(true) {
    P(odd);
    P(mutex);
    getodd();
    V(mutex);
    V(empty);
    countodd();
 }
}

void P3() {
 while(true) {
    P(even);
    P(mutex);
    geteven();
    V(mutex);
    V(empty);
    counteven();
 }
}
```

2.

```
semaphore empty = M;    // empty=0 表示空锅
semaphore full = 0;     // full=M 表示满锅
semaphore mutex = 1;    // 保护对 S 的访问
int S = 0;              // 锅中肉的份数

void savagery(){
    while (true){
        P(full);
        P(mutex);
        S = S - 1;
        V(mutex);
        V(empty);
        getServingFromPot();
        eat();
    }
}

void chef(){
    int i;
    while(true) {
        for (int i =0;i < M;i++)
            P(empty);
        P(mutex);
        S = S + M;
        V(mutex);
        for(int i =0;i < M;i++)
            V(full);
        putServingsInPot(M);
    }
}
```

3.

```
semaphore empty = 1000;          // empty = 0 缓冲区空
semaphore full = 0;              // full = 1000 缓冲区满
semaphore only = 1;              //保护只有一个消费者线程在取产品
semaphore mutex = 1;             //保护对 num 的访问
int num = 0;                     // 缓冲区产品数

void producer() {
 while(true) {
    product = produce();
    P(empty);
    P(mutex);
    num++;
    V(mutex);
    V(full);
    insert_product(product);
 }
}

void customer() {
 int i;
 while(true) {
    P(only);
    for (i = 0;i < 10;i++) {
        P(full);
        P(mutex);
        num--;
        V(mutex);
        V(empty);
        custome_product();
    }
    V(only);
 }
}
```