题目一解答

语句顺序交换后，进程i、j的示意代码如下：

```
/*
  进程j的示意代码
*/
turn = j;
flag[i] = true;

while(flag[j] && turn==j)
{
  /* waiting */
}
/* 进入临界区 */

flag[i] = false;

/* 退出临界区 */
```

```
/*
  进程i的示意代码
*/
turn = i;
flag[j] = true;

while(flag[i] && turn==i)
{
  /* waiting */
}
/* 进入临界区 */

flag[j] = false;

/* 退出临界区 */
```

互斥条件不满足；

当按如下顺序执行时：

```
turn = j;//Process i
/*上下文切换至Process j*/
turn = i;//Process j
flag [j] = true;// Process j
while(flag[i] && turn == i)//Process j,判断不成立，此时flag[i]==false，进入临界区
/*Process j 进入临界区*/
/*上下文切换至Process i*/
flag[i] = true;//Process i
while(flag[j] && turn == j)//Process i,判断不成立，此时turn==i，进入临界区
/*Process i 进入临界区*/
```

满足有空让进和有限等待；

题目二解答

在单处理器系统中，使用自旋锁将导致进程在等待锁的同时还在运行（忙等），这样虽然避免了上下文切换的开销（通常上下文需要花费比较长的时间），但是却让CPU周期浪费在忙等上，不能被其他进程所利用；

而在多处理器系统中，一个这样的线程在处理器上自旋时，另一线程可以在其他处理器上在其临界区执行，这样即避免了上下文切换的开销，也让其他进程能够利用CPU周期。

题目三解答

```
/*互斥锁包含的数据结构及函数*/
typedef struct{
    int available;
}lock;
void acquire(lock *mutex);
```

```
void release(lock *mutex);

/*实现互斥锁机制*/
void acquire(lock *mutex)
{
    if(mutex->available == 1)
    {
        add this process to wait list/*加入等待队列*/
        block();
    }
}
void release(lock *mutex)
{
    compare_and_swap(&mutex->available,1,0);
    remove a process P form wait list/*从等待队列中删除Process P*/
    wakeup(P);
}
```

其中，compare_and_swap(&mutex->available,1,0)表示如果available的值是1，就把它改成0，然后返回值为1；

题目四解答

```
/*理发师、顾客共享以下数据结构*/
semaphore customer=0; //顾客数
semaphore avai=1;//当前可服务人数，为负时，绝对值表示等待人数
semaphore barber=0;//barber<0时，理发师在睡觉ing
int wait_num=0;//前面等待人数

/*理发师进程*/
do{
    wait(barber);//睡觉中....
    while(wait_num!=0)
    {
        if(wait_num==1)//第一个
        {
            //理发....
            //理发结束
        }
        else//需要叫顾客来剪头
        {
            signal(customer);//叫顾客
            //理发....
            //理发结束
        }
        wait_num--;
        signal(avai);
    }
    //没人了，下个循环将开始睡觉，等待被再次叫醒
}while(true)

/*顾客进程*/
//顾客剪完头就离开了，不需要循环执行
{
    wait(avai);
    if(wait_num==0)//第一个来
    {
        wait_num++;
        signal(barber);//第一个顾客负责叫醒理发师
```

```
    }
    else if(wait_num<=n)//后面来的顾客，等待....
    {
        wait_num++;
        wait(customer);//挂起自己，等待被理发师叫
    }
    else if(wait_num>n)//没有座位了，取消剪发的念头
    {
        signal(avai);
        /*离开*/
    }
}
```