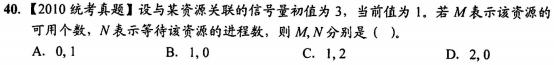
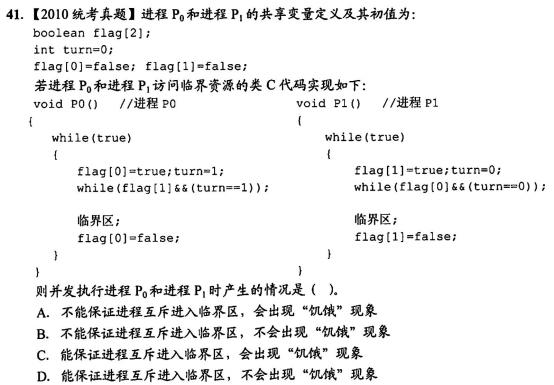
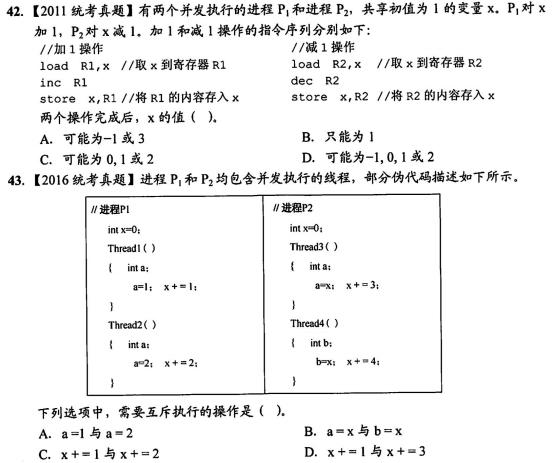
**同步与互斥**



B

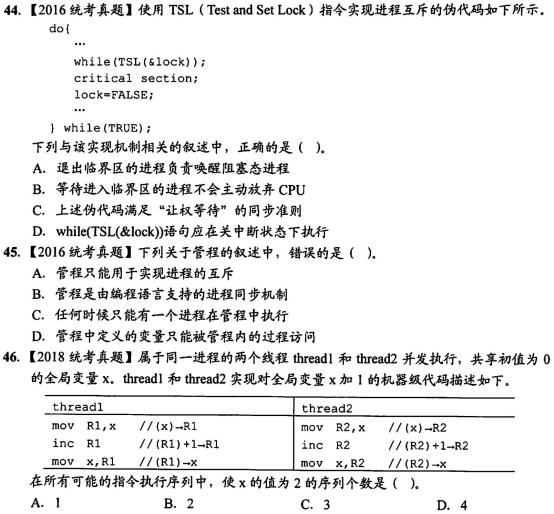


D



C

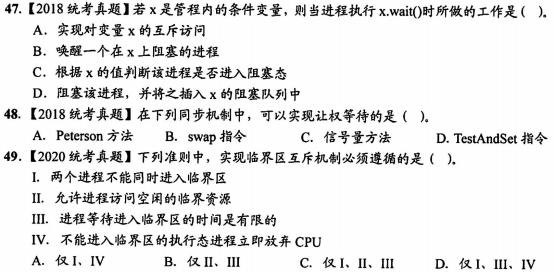
C



B

A

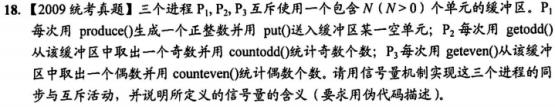
B



D

C

C



答：

semaphore mutex = 1 // 用于缓冲区的互斥访问

semaphore empty = N // 表示缓冲区的空单元数，初始为N

semaphore odd = 0 // 表示缓冲区中奇数的个数

semaphore even = 0 // 表示缓冲区中偶数的个数

int count\_odd = 0 // 统计奇数个数

int count\_even = 0 // 统计偶数个数

// P1，生产正整数并放入缓冲区

process P1 {

while (true) {

int item = produce();

P(empty);

P(mutex);

put(item);

V(mutex);

if (item % 2 == 0) {

V(even);

} else {

V(odd);

}

}

}

// P2，从缓冲区取奇数并统计

process P2 {

while (true) {

P(odd);

P(mutex)

int item = getodd();

count\_odd = countodd();

V(mutex);

V(empty);

}

}

// P3，从缓冲区取偶数并统计

process P3 {

while (true) {

P(even);

P(mutex);

int item = geteven();

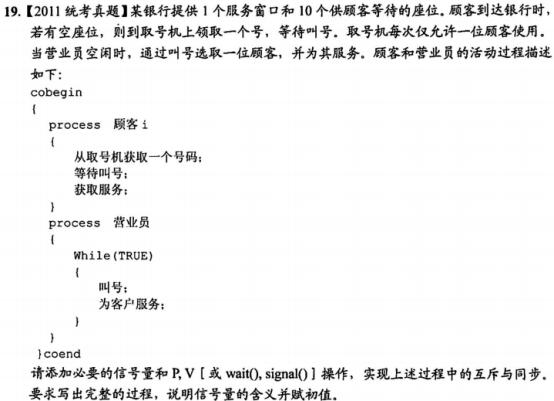
count\_even = counteven(c);

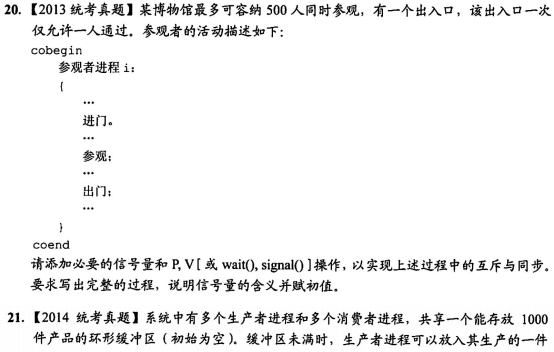
V(mutex);

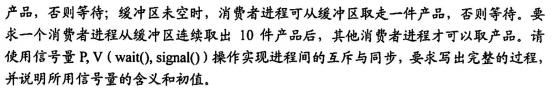
V(empty);

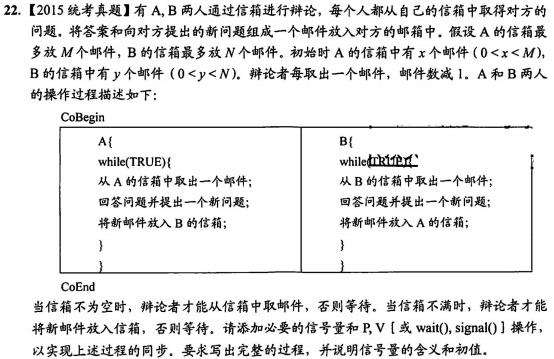
}

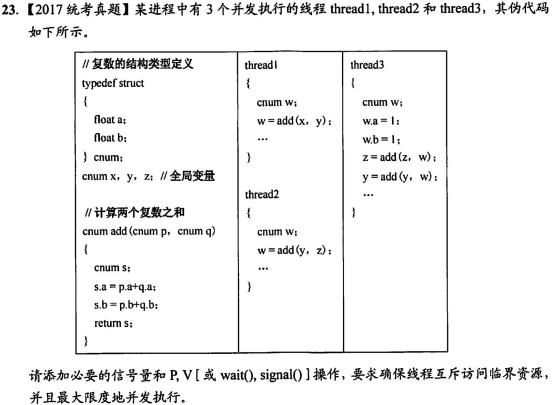
}

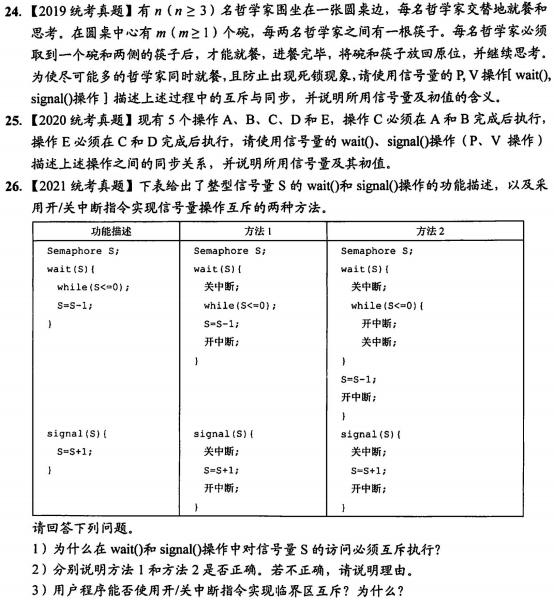




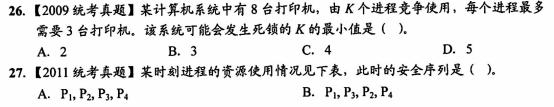






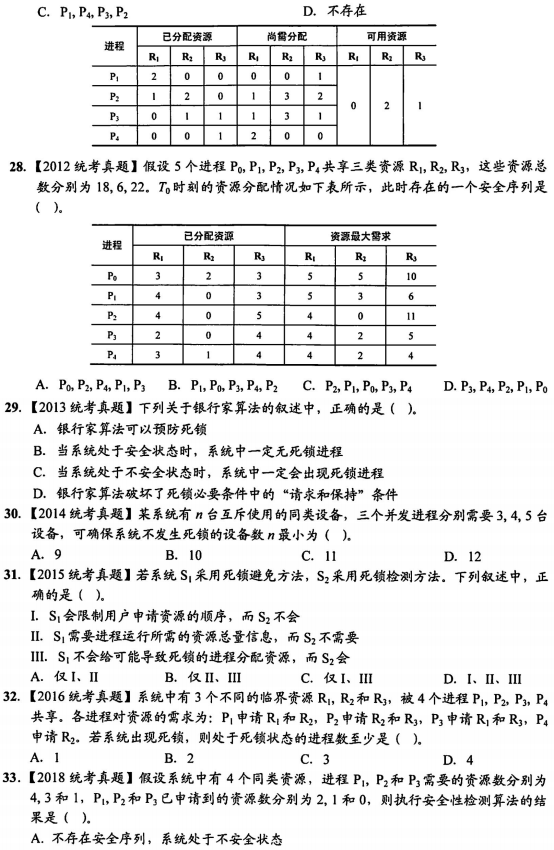


**死锁**



C

D

D

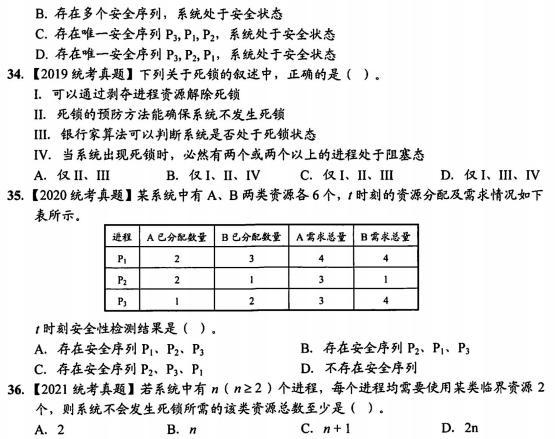
B

B

B

C

A

B

B

C