1. 设信号量 empty 表示水缸还能容纳几桶水，初值为 10；

设信号量 full 表示水缸中还有几桶水可用，初值为 0；

设信号量 S 表示可用水桶数目，初值为 3;

设信号量 mutex1 表示从井中打水者互斥使用水井，初值为 1；

设 mutext2 表示互斥使用水缸，初值为 1。

程序如下：

```
Cobegin
  小和尚打水：                          老和尚取水：
  begin:                               begin:
   L1: P(empty);                        L2: P(full);
       P(S);                                P(S);
       P(mutex1);                           P(mutex2);
       井中打水；                           从缸中取水；
       V(mutex1);                           V(mutex2);
       P(mutex2);                           V(S);
       倒水入缸；                           V(empty);
       V(mutex2);                           goto L2;
       V(S);                               end
       V(full);
       goto L1;
   end
Coend
```

2. 信号量的定义和初值分别如下：

```
semaphore full_A=x;          //表示 A 的信箱中的邮件数量
semaphore full_B=y;          //表示 B 的信箱中的邮件数量
semaphore empty_A=M-x;       //表示 A 的信箱中还可存放的邮件数量
semaphore empty_B=N-y;       //表示 B 的信箱中还可存放的邮件数量
semaphore mutex_A=1;         //用于对 A 的信箱进行互斥操作
semaphore mutex_B=1;         //用于对 B 的信息进行互斥操作
```

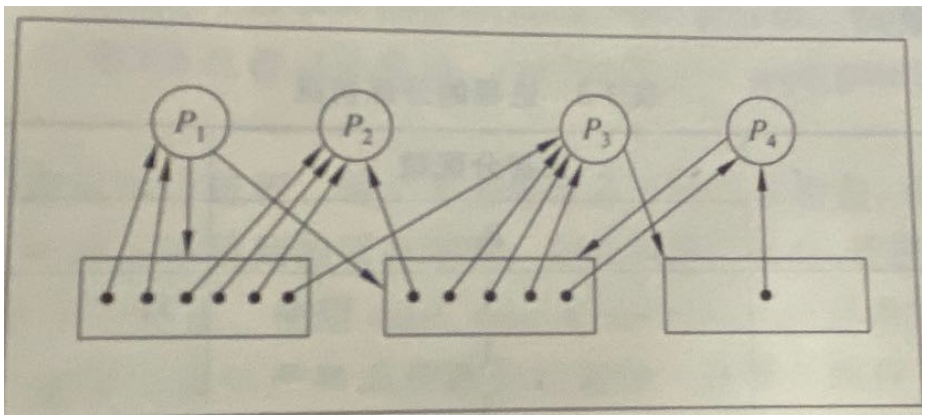
程序如下：

```
CoBegin
  A{                                    B{
    while(TRUE) {                         while(TRUE) {
      P(full_A);                            P(full_B);
      P(mutex_A);                           P(mutex_B);
      从 A 的信箱中取出一个邮件;              从 B 的信箱中取出一个邮件;
      V(mutex_A);                           V(mutex_B);
      V(empty_A);                           V(empty_B);
      回答问题并提出一个新问题;              回答问题并提出一个新问题;
      P(empty_B);                           P(empty_A);
      P(mutex_B);                           P(mutex_A);
      将新邮件放入 B 的信箱;                 将新邮件放入 A 的信箱;
      V(mutex_B);                           V(mutex_A);
      V(full_B);                            V(full_A);
    }                                     }
  }                                     }
CoEnd
```

3. (1) 3 种同步方式：发送进程 P 阻塞，接收进程 Q 阻塞；发送进程 P 不阻塞，接收进程 Q 阻塞；发送进程 P 不阻塞，接收进程 Q 不阻塞。

(2) 发送和接收均阻塞：X 的值是 11；无阻塞发送和阻塞接收：X 的值是 11 或 21；无阻塞的发送和接收：X 的值可能是 11、21 或-99。

4. (1) 资源分配图如下：



(2) 简化步骤如下：

(a) Round 1, Step 1: 所有资源都有未满足的资源请求，不能化简；

(b) Round 1, Step 2: 进程 P2 所有的资源请求均可满足，擦除相应连线；

(c) Round 1, Step 3: 进程 P1 对资源 R1 和 R2 的请求可以满足，擦除相应连线；

(e) Round 2, Step 1:资源 R1 能够满足所有的资源请求，擦除相应连线；

(f) Round 2, Step 2: 进程 P1 所有的资源请求均可满足，擦除相应连线；

(g) Round 2, Step 3: 进程 P4 对资源 R2 的请求可以满足，擦除相应连线；

(h) Round 3, Step 1:资源 R1 和 R2 能够满足所有的资源请求，擦除相应连线；

(i) Round 3, Step 2: 进程 P4 所有的资源请求均可满足，擦除相应连线；

(j) Round 3, Step 3: 进程 P3 对资源 R3 的请求可以满足，擦除相应连线；

(3) 不会。经过第 2 步化简后，资源分配图中没有任何连线，意味着不会发生死锁。