4.现有5条语句：

S1：a=5-x；

S2：b=a*x；

S3：c=4*x；

S4：d=b+c；

S5：e=d+3.

试用Bernstein条件证明语句S2和S3可以并发执行，而语句S3和S4不可并发执行。

Bernstein条件

定义：

R(Pi)={a1，a2，…，an}　读集

W(Pi)={b1，b2，…，bn}　写集

若两个进程满足Bernstein条件，即

R(Pi) ∩W(Pj) ∪ R(Pj) ∩W(Pi)

∪ W(Pi) ∩W(Pj)=ϕ

那么这两个进程可以并发执行。

解：

R(S2)={a,x}　　W(S2)={b}

R(S3)={x}　　W(S3)={c}

R(S4)={b,c}　　W(S4)={d}

R(S2) ∩W(S3) ∪ R(S3) ∩W(S2)
∪ W(S2) ∩W(S3)=ϕ

所以S2和S3可以并发执行。

R(S3) ∩W(S4) ∪ R(S4) ∩W(S3)
∪ W(S3) ∩W(S4)={c} ≠ ϕ

所以S3和S4不能并发执行。

6.在一个盒子里，混装了数量相同的黑白围棋子。现利用自动分拣系统把黑子、白字分开，设分拣系统有两个进程P1和P2，其中P1拣白子，P2拣黑子。规定：每个进程每次拣一子；当一个进程在拣时，不允许另一个进程去拣；当一个进程拣一子时，必须让另一个去拣。试写出进程P1和P2 能够正确并发的程序。

两个进程的同步问题

解：设信号量s1和s2分别表示可拣白子和黑子，规定先拣白子。

```
var s1,s2: semaphore;
s1:=1;  s2:=0;
Cobegin
 {  Process  p1                    Process  p2
    begin                          begin
        repeat                        repeat
        P(s1);                        P(s2);
        拣白子;                       拣黑子;
        V(s2);                         V(s1);
        until false;                   until false;
    end                            end
                                 } coend
```

28.设当前的系统状态如下：系统此时Available=（1,1,2）

	Claim			Allocation		
进程	R1	R2	R3	R1	R2	R3
P1	3	2	2	1	0	0
P2	6	1	3	5	1	1
P3	3	1	4	2	1	1
P4	4	2	2	0	0	2

（1）计算各进程需要的资源数Cki – Aki

（2）系统是否处于安全状态，为什么？

（3）P2发出请求向量request2（1,0,1），系统能把资源分给它吗？

（4）若在P2申请资源后，若P1发出请求向量request1（1,0,1），系统能把资源分给它吗？

（5）若P1申请资源后，若P3发出请求向量request3（0,0,1），系统能把资源分给它吗？

解：（1）P1、P2、P3、P4所需要的资源数Cki – Aki如下表所示：

	R1	R2	R3
P1	2	2	2
P2	1	0	2
P3	1	0	3
P4	4	2	0

（2）系统处于安全状态，存在安全序列（P2，P1，P3，P4)。

（3）request2（1,0,1）

{ Request2（1,0,1）<Ck2 - Ak2=(1,0,2)
  Request2(1,0,1)<Available(1,1,2)

系统尝试为P2分配资源，分配后系统状态如下：

	Allocation			Cki – Aki			Available		
进程	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	1	0	0	2	2	2	0	1	1
P2	6	1	2	0	0	1			
P3	2	1	1	1	0	3			
P4	0	0	2	4	2	0			

安全性分析表：

	Currentavail	Cki – Aki	Allocation	Currentavail +Allocation	Possible
	R1 R2 R3	R1 R2 R3	R1 R2 R3	R1 R2 R3	
P2	0 1 1	0 0 1	6 1 2	6 2 3	True
P1	6 2 3	2 2 2	1 0 0	7 2 3	True
P3	7 2 3	1 0 3	2 1 1	9 3 4	True
P4	9 3 4	4 2 0	0 0 2	9 3 6	True

可见可以找到一个安全序列（**P2，P1，P3，P4**），所以可以为**P2**分配资源。

（4）Available（0,1,1）<request1(1,0,1)

资源不足，所以不能分配。

（5）先进行安全性检

{ Request3（0,0,

Request3（0,0,

试分配后，系统状态如下所示：

此时剩余的资源以不能满足任何进程的需求，故系统以处于不安全状态，所以不能把资源分配给P3。

	Allocation			Cki – Aki			Available		
进程	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	1	0	0	2	2	2			
P2	6	1	2	0	0	1			
P3	2	1	2	1	0	2	0	1	0
P4	0	0	2	4	2	0			