Q1.

a.p1, p2 (writes 0), q1, q2, p2 (writes 2), p1 (exits).

b.q1, q2, p1 produces no output, and both halt.

c.The value 2 can appear only once, because the value of n can only increase. So if p2: write(n)

is executed with n = 2, the subsequent execution of p1: while n < 2 prevents another execution

of p2.

d.q1 followed by (p1, p2)* but fairness says q2 has to run sometime, so then p1 exits. So 1 can appear a finite number of times.

Question 2.

(a) See textbook, p. 152.

(b) Three, one for each condition variable, and one for monitor entry.

(c) Like (a), but remove condition variables. Guard entry â€œputâ€□ by

â€œwhen buffer not fullâ€□ and entry â€œgetâ€□ by â€œwhen buffer not emptyâ€□.

The bodies of the entries need no â€œifâ€□ or â€œsignalâ€□.

Question 3.

(Part a)

The table is:

	State=(pi, qi, S)	next state if p moves	next state if q moves
1.	(p2, q2, Z)	(p3, q2, P)	(p2, q3, Q)
2.	(p2, q3, Q)	(p3, q3, QP)	(p2, q5, Q)
3.	(p2, q5, Q)	(p3, q5, QP)	(p2, q2, Z)
4.	(p3, q2, P)	(p5, q2, P)	(p3, q3, PQ)
5.	(p3, q3, PQ)	(p5, q3, PQ)	no move
6.	(p3, q3, QP)	no move	(p3, q5, QP)

```
7.  (p3, q5, QP)        no move                 (p3, q2, P)

8.  (p5, q2, P)         (p2, q2, Z)             (p5, q3, PQ)

9.  (p5, q3, PQ)        (p2, q3, Q)             no move
```

(Part b) There is no state with (p5, q5, S)

(Part c) Every state has at least one move.

(Part d).

Prove that every p2-state leads to a p5-state.

By definition, 8 and 9 are in M.

5 has to lead to 9. So M = {5, 8, 9}

4 leads to 8 or 5.   So M = {4, 5, 8, 9}

6 and 7 have to lead to 4. So M = {4, 5, 6, 7, 8, 9}

1-2-3 can loop by doing only q moves. By fairness, one of them must do
a p step at some point,

leading to 4, 6, or 7.

So M = {1, 2, 3, 4, 5, 6, 7, 8, 9}.

Question 4.

Let Mp = p4- (S=P v S=PQ).

4a.   Show that Mp is invariant.

If !p4, Mp holds.

Can p arrive at p4 when the consequent is false?

p4 can only happen if p does p3.

Then the await ensures (S = P v S = PQ).

Suppose p is at p4, the consequent is true, and then q spoils it.

But only q2 and q5 assign to S.

(S = P v S = PQ) followed by q2 yields S=PQ.

(S = P v S = PQ) followed by q5 also yields S=P v S=PQ.

So []Mp

```
4b.  Show mutex,

Mq  = q4- (S=Q v S=QP).

p4 & q4 - false  (no agreement on S).

So no such state exists.

----------------------------

4c.  Show that there cannot be deadlock.

Suppose [](p3 & q3).  Then both awaits wait.  So S=Z.

But after p2 or q2 (one of them must be the last thing before p3&q3),

S cannot remain Z, and neither produces Z.

So S is P or QP or Q or PQ.

So [](p3 & q3) is impossible.


-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
---

Question 5. The thing to realise is that W carries a charge of +2, G
carries a charge of +1, and B a charge of -1.    Then val is the excess
charge.

There is a conservation of charge, because the values w, b, g change
only in lines 15 and 20.  Line 15 follows removal of W and B and posting
of G, so it reflects the

charge correctly.  Line 20 follows removal of G and B.

The computation continues until only the excess charge remains, all
balancing charges having been removed.  That is what lines 33 and 34 do.

a. MaxW=MaxB=3 means val = 2*3 -  3 = 3.   So the only possibilities are
w=1, g=1 or g=3.

Remove (W), Remove(B), Post(G) thrice, i.e., W+B - G,  W+B - G, W+B-G
gives the latter.

Any process could do this, because the W has to be removed first, and
the B cannot be stolen by someone without a B.

Do Remove(W), Remove(B), Post(G), Remove(W), Remove(B), Post(G).
Interpolate Remove(G), Remove(B) sequences.  This gives w=1, g=1.

b. Val is invariant.  See above.

c. Termination.  See above.

d. Consider MaxW=MaxB=1.  The only result should be W+B-G.  But with two
Râ€™s, we could have R1 taking W and R2 taking B, followed by both timing
```


out and replacing the W and B. Loop.

The whole question illustrates how to avoid deadlock/livelock in resource allocation by imposing an order on the resources. There is no need for synchronization between the Rs, so the Linda is almost irrelevant.

Question 6.

a. Write I for Ints, E for End, and C(c) for Cell holding c. Show links with the name of the channel between daxhes.

I â€“qinit- E

After 34,

I â€“qinit- C(34) â€“q- E

After 76

I â€“qinit- C(34) â€“qout- C(76) â€“q- E

After 23

I â€“qinit- C(23) â€“q- C(34) â€“qout- C(76) â€“q- E. We might want to use notation or colours to show whose local channel goes where.

The program as a whole does an insertion sort, printing out the sorted list as a 0 goes through the chain of processes.

b. CAP has no effect.

c. Add the line

```
n == c- run Cell (c, qin, qout)
```

as one of the branches of the if in Cell.

d.

```
 :: n==0  - printf("%d\n", c);
    qout ! 0;
    run Cell (c, qin, qout)
```

is the change to be made to the branch n==0.