1. Answer:

No. Consider the following case.

1. Initially `turn` equals 0.
2. P(1) sets `blocked[1]` to true (line 6), enters the 2nd while loop (line 7), and finds `blocked[0]` false, so skips the 3rd while loop (line 8). Next, P(1) switches out and P(0) switches in.
3. P(0) then sets `blocked[0]` to true (line 6), finds `turn` = 0 (line 7), and enters its critical section (line 12). Next, P(0) switches out and P(1) switches in.
4. P(1) then assigns 1 to `turn` (line 10) and also enters its critical section (line 12).

This "solution" was invented, submitted to a scholarly journal, checked by the editor and one or more referees, and none of them realized the algorithm was flawed. The algorithm was published in the journal and it had to be "retracted" when it was later discovered to be incorrect!

(Harris Hyman, Comments on a problem in concurrent programming control, Communications of the ACM, v.9 n.1, p.45, Jan. 1966)

2. Answer:

It doesn't work. Consider the following case.

- After the producer appends the ***first*** data item into an empty buffer (n=1), calls semSignal (delay=1) and leaves the critical section, the producer switches out and the consumer switches in to take the ***first*** data item out of the buffer (delay=0 and n=0).
- Right after the consumer leaves the critical section, the producer switches in to append the ***second*** data item (n=1) and calls semSignal (delay=1).
- Right after the producer leaves the critical section, the consumer switches in and resumes to check the value of n=1 and FAILS to call SemWaitB.
- The consumer continues to take the ***second*** data item out of the buffer (n=0).
- Even though the consumer calls SemWait, it does not wait because delay was 1.
- As a result, the consumer can continue to take and consume a non-existing data item (n= -1)!

3. Answer:

a) `wsem` = -1, `rsem` = 0, `x` = 1, `y` = 1, `z` = 1,

b) `z` = 0, `rsem` = -1, others remain no change

c) `z` = -1, others remain no change

d) `wsem` = -2, others remain no change

e) the first writer

Self-test

1. C
2. C
3. A
4. semWait and semSignal
5. A
6. B