Problem 1

a)
1. T1 is followed by T2:(A,B)=(0,10)
2. T2 is followed by T1:(A,B)=(20,0)

Hence, there are two possible pairs of values (A,B) can get in serial schedules:
(0,10) and (20,0)

b)
Possible pairs of values (A,B): (10,20), (0,10), (20,0)

c)
T1:
```
lock-S(A);
lock-X(B);
read(A);
read(B);
if(A==0) then B=B+10;
write(B);
unlock(A);
unlock(B);
```

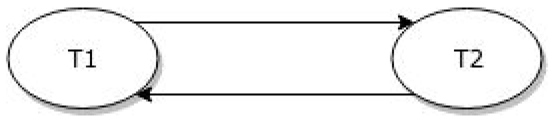
T2:
```
lock-X(A);
lock-S(B);
read(B);
read(A);
if(B==0) then A=A+20;
write(A);
unlock(A);
unlock(B);
```

d)
Possible pairs of values (A,B): (0,10) (20,0)

e)
Yes, the schedule shown below can result in deadlock.
```
T1                    T2
lock-S(A);
                      lock-S(B);
lock-X(B);
                      lock-X(A);
read(A);
read(B);
if(A==0) then B=B+10;
write(B);
unlock(A);
unlock(B);
                              read(B);
                              read(A);
                              if(B==0) then A=A+20;
                              write(A);
                              unlock(A);
                              unlock(B);
```

Problem 2

a)



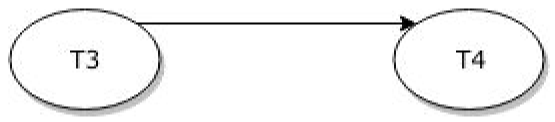
b)
The schedule is not conflict serializable, for the precedence graph has a cycle.

c)
No.
Any schedule produced by 2PL protocol has an acyclic precedence graph, but the precedence graph of this schedule has a cycle.

d)
No.
Since T1 writes A before T2 reads A, T1 has to commit before T2. However, T2 writes B before T1 reads B, which requires that T2 has to commit before T1. Since T1 and T2 cannot commit at the same time, there are no places where commit statements can be placed.

Problem 3

a)



b)
The schedule is conflict serializable.

c)
Yes.

```
T3:                     T4:
lock_X(A)
lock_X(A) granted
lock_X(B)
lock_X(B) granted
read A
A:=A+1
write A
unlock(A)
                        lock_X(A)
                        lock_X(A) granted
                        read A
                        A:=A+1
                        write A
read B
B:=B-1
write B
unlock(B)
                        lock_X(B)
                        lock_X(B) granted
                        unlock(A)
                        read B
                        B:=B-1
                        write B
                        unlock(B)
```

d)
Yes.

```
T3:                     T4:
```

```
read A
A:=A+1
write A
                    read A
                    A:=A+1
                    write A
read B
B:=B-1
write B
commit
                    read B
                    B:=B-1
                    write B
                    commit
```