1. There was concurrency long before there were databases. BamBam and Pebbles were two cave-
people who liked to carve pictures on stones together. Pebbles liked to copy BamBam's pictures,
but if she couldn't see his picture she would carve a picture of a saber-tooth tiger. Both Pebbles
and BamBam always smashed each others' pictures to bits as a sign of affection.

Consider the following sequence of events:
B1) BamBam carves a picture of a stegasaurus on his stone
P1) Pebbles sees BamBam's picture
P2) Pebbles carves a picture of a stegasaurus on her stone.
B2) BamBam smashes Pebbles' stone
P3) Pebbles smashes BamBam's stone.

For the purposes of this question, assume that B and P are “transactions” in the ACID sense, and
that both commit at the end of that sequence of events.

a. Rewrite the sequence of events as a schedule with two transactions:

<table>
<thead>
<tr>
<th>Time</th>
<th>B: W(S1)</th>
<th>W(S2)</th>
<th>Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>P:</td>
<td>R(S1)</td>
<td>W(S2)</td>
<td>W(S1)</td>
</tr>
</tbody>
</table>

b. Draw a dependency graph for the schedule of part (a).

```
B --- P
```

c. Is this schedule conflict serializable (Y/N)? (No explanation is necessary.)

N

d. Is this schedule serializable (Y/N)? (No explanation is necessary.)

Y (Also accepted per regrade discussion: N)

e. Is this schedule view serializable (Y/N)? (No explanation is necessary.)

N
2. Consider the following schedule, which was truncated at the right:

<table>
<thead>
<tr>
<th>T1: R(X) W(X) Abort</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2: R(X) R(Y) W(Y)</td>
</tr>
</tbody>
</table>

a. Which locking protocol(s) that we studied would allow this schedule? (Circle one)
   i. Two-phase locking
   ii. Strict two-phase locking
   iii. Intent locking
   iv. Both (i) and (iii) allow it Also accepted.
   v. All of the above allow it

b. What phenomenon that we studied will occur after T1 Aborts?
   
   Cascaded abort. Partial credit for just mentioning that T2 aborts.

3. Consider the following SQL query:

   UPDATE Students SET final_grade = 'A' WHERE studentID = 314159;

   Assume there are no indexes on Students. List all locks that should be acquired so that the query can be executed while maintaining consistency and allowing as much concurrency as you can (preventing as few other transactions from continuing as possible). For each lock listed in this way, state the object that is to be locked and the lock mode (for example, an exclusive-mode lock on object "A" would be written as "X(A)").

   SIX(Students)
   X(tuple with studentID=314159)
4. Consider the following sequence of lock requests from two transactions:

T1: S(A) X(B) \hspace{1cm} S(A) X(B)
T2: S(B) X(A)

Also accepted:
T1: S(A) X(B)
T2: S(B) X(A) \hspace{1cm} S(B) X(A)

5. 

a. Assume these lock requests were being attempted in a system with strict two-phase locking and deadlock detection. Draw the waits-for graph for this schedule.

![Waits-for graph]

b. Extend the lock-sequence timeline above to illustrate how the two transactions can eventually execute to completion in a system with two-phase locking and deadlock-detection. You do not need to add anything to the sequence except new lock requests.

   See above

c. Assume the original sequence of requests above was attempted in a system with strict two-phase locking, and deadlock prevention with a “wound-wait” policy. T1 has higher priority than T2. At what point in the schedule would the transactions be prevented from continuing? What would happen to the schedule from then onward?

   When T1 requests an X lock on B, it would “wound” (abort) T2 and acquire the X lock on B. T2 would then restart. T2 could not then acquire the S lock on B until T1 committed.

   T1: S(A) X(B)
   T2: S(B) S(B) X(A)
6. Consider the following schedule that occurs in a system with Optimistic Concurrency Control:

\[
\begin{align*}
T1: & \quad R(B) \quad R(A)W(A) \\
T2: & \quad R(A) \quad R(B) W(B)
\end{align*}
\]

Assume that T1 finished its read phase and its validation stage first, and is in the midst of its write phase when T2 enters its validation phase. Will T2 pass the validation phase? Justify your answer.

NO

Test 1 clearly fails due to the overlapping validation/write. Tests 2 & 3 fail because WS(T1) intersect RS(T2) is not empty.