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:
- BamBam carves a picture of a stegasaurus on his stone
- Pebbles sees BamBam's picture
- Pebbles carves a picture of a stegasaurus on her stone.
- BamBam smashes Pebbles' stone
- 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 reading and writing two stones (SB and SP).

<table>
<thead>
<tr>
<th>Time</th>
<th>B:</th>
<th>P:</th>
</tr>
</thead>
</table>

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

c. Is this schedule conflict serializable (Y/N)? (No explanation is necessary.)
d. Is this schedule serializable (Y/N)? (No explanation is necessary.)
e. Is this schedule view serializable (Y/N)? (No explanation is necessary.)
2. Consider the following schedule, which was truncated at the right:

<table>
<thead>
<tr>
<th>T1:</th>
<th>R(X)</th>
<th>W(X)</th>
<th>Abort</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2:</td>
<td>R(X)</td>
<td>R(Y)</td>
<td>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
v. All of the above allow it

b. What action will the DBMS take after T1 Aborts? What is the general name of this phenomenon?

3. Consider the following SQL query:

```sql
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)").
4. Consider the following sequence of lock requests from two transactions:

T1: S(A)     X(B)
T2:     S(B)     X(A)

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.

b. Extend the lock-request 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 timeline except new lock requests.

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 some point in the lock-request timeline above the transactions would be unable to proceed with their lock requests, and the remainder of the timeline would be different. Write down the lock-request timeline that would occur under strict two-phase locking and wound-wait, with T1 higher-priority than T2.

5. Consider the following schedule that occurs in a system with Optimistic Concurrency Control:

T1: R(B)     R(A) W(A)
T2: R(A)     R(B) W(B)

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.