Question 1. [30 points] Short Questions

a) **[7 points]** Illustrate one example of a query plan that the System R optimizer (Selenger'79) will not consider and is a plan that is more efficient that the plans that the System R optimizer will consider?

b) **[8 points]** Consider the B-link protocol. What is the maximum number of latches that can be acquired by a transaction? Illustrate the scenario where this maximum number of latches is acquired.

c) **[7 points]** 2PC with Presumed Commit is better (higher performant) than 2PC with Presumed Abort for read-only transactions.

☐ True ☐ False

Justify your answer

d) **[8 points]** [5 points] Queries that are submitted to a traditional database system simply consist of the (SQL) query statement. However, queries that are submitted to the Mariposa system include one additional piece of input information. What is this additional information?

Question 2. [30 points] Concurrency Control

Consider the following two statements about the Granularity of Locks paper by Gray et al.:

1. Before requesting an S or IS lock, **ALL** paths to the root must be locked in IS mode or greater, and
2. Before requesting an IX, SIX, or X lock, **ANY** path to the root must be locked in IX mode or greater.

a) **[5 points]** There is something different about these two statements when compared to those in the paper. What is wrong?

b) **[15 points]** If Gray et al. had used these two statements instead, would the protocol still produce a serializable ordering (presuming we are well formed wrt reads and writes, and have predicate locks for phantom protection)? Do you expect transactions to complete at the same rate, or are the Gray et al. statements superior to those mentioned in this question? If they behave differently, provide one or more examples to show when.

c) **[10 points]** In the Optimistic Concurrency Control paper, Kung and Robinson discuss a problem with long running transactions that feature long read phases. Such a transaction T might need unbounded buffer space to keep track of the write sets of all transactions that have begun and finished while T was active. The stated solution in the paper is to bound the buffer space and just abort/restart such transactions. This can lead to starvation, which can lead to a need to lock all other xacts out of the system while T runs.

Suggest an alternate solution that does not require aborting transaction T but that also allows us to bound buffer space in memory. Remember that this paper was written in 1981 and we did not have large, fast storage devices below RAM in the memory hierarchy. Use this fact in the design of your solution, and try to stay in memory if possible.

Question 3. **[20 points] Recovery**

a) **[6 points]** Consider the four figures below that indicate log records being processed by ARIES during recovery after a crash. Each dot on the timeline represents an update log record. In the log records assume that there is an update record right before and after the checkpoint record. The arrows denote the direction and the position from which the logs are scanned during the three phases of recovery. The arrow labeled A, R, and U denote the Analysis, Redo and Undo phases respectively. Which of these diagrams represent processing that can **never** occur in ARIES? Why?

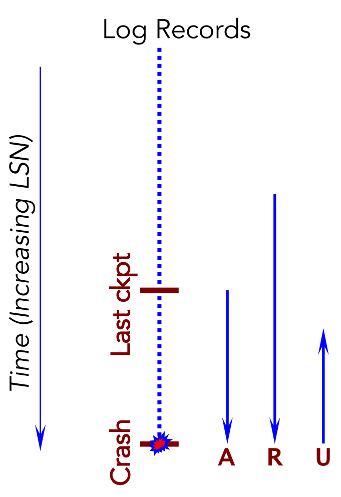

Figure (a)

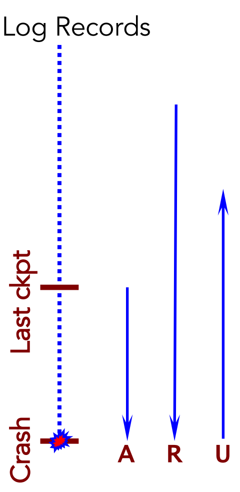

Figure (b)

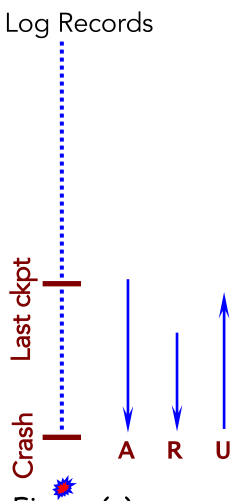

Figure (c)

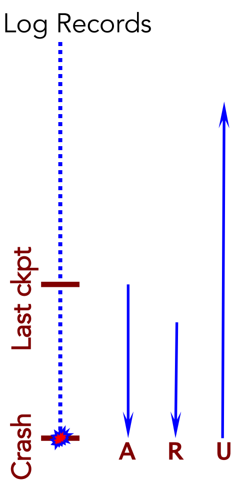

Figure (d)

Answer: ____________

Why?:

b) **[14 points]** Recall that updates to indexes are *logically* logged (e.g. insert record with key 5, delete record with key 92) instead of *physically* logged (e.g. making a copy of the index pages before and after the insert of 5 and delete of 92). Consider a user's desire to convert a heap (normal row storage) table into a clustered index-based table (with the full records in the leaf level) with the index attribute(s) forming a unique key. What is an efficient method of converting the table? Describe the technique and its effects on logging, locking, and recovery if a system crash occurs.