The following is the source for question 1:
“Database Management Systems.” *Chegg Study*, Chegg, www.chegg.com/homework-help/database-management-systems-3rd-edition-chapter-16-solutions-9780072465631.

1.1 A transaction is defined as anyone execution of a user program in a DBMS. A transaction can be seen by the DBMS as a list or series of actions. Reads are writes as the actions a transaction can execute. In an ordinary program actions could involve user input, access to network devices etc.

1.2
Atomicity: a transaction executes when either none or all of the actions of the transactions or fully compete. There are no partial transactions.
Consistency: beginning and finishing a transaction with a ‘consistent’ database. For example, without the appropriate deposit or withdrawals, money in a bank database should never be created or deleted.
Isolation: ensures a transaction can be run without considering side effect at a transaction running concurrently might have (a transaction can run independently). The database project transactions from the effects of other transaction when it interleaves transaction for performance reasons.
Durability: persistence of commited data. Even if the system crashes before the data is written to non-volatile storage, if a transaction was committed it should persist in the database.
Schedule: Series of transactions that are possibly overlapping.
Blind write: when a transaction white an object without ever reading that object.
Dirty read: occurs when a database object that has been modified by a not-yet-committed transaction is read in a another transaction.
Unrepeatable read: when a transaction cannot read the same object value multiple times even though the transactions has not modified the value. For example, suppose the value of an object A is modified by a transaction T2 but A has been ready by a transaction T1 that is still in progress. T1 will get a different result if it tries to read the value of A again even though it has not modified A.
Serializable schedule: is a schedule whose effect on any consistent database instance over a set S of transactions is identical to that of a complete serial schedule of the set of committed transactions in S.
Recoverable schedule: a schedule in which a transaction can only commit if all the other transaction whose changes it has read have committed.
Avoids-cascading-aborts schedule: transactions only read changes of committed transaction. This type of schedule can abort a transaction with cascading the abort to other transactions. It is also recoverable.

1.3
Strict 2PL: most widely used locking protocol where

1) Before a transaction reads/modifies an objects it requests a shared/exclusive lock on the object
2) When a transaction is complete, all locks held by the transaction are released.

1.4
The phantom problem occurs when a transaction, which follows the strict 2PL protocol and does not modify any of these objects itself, retrieves a collection of objects twice but sees different results. The phantom problem normally occurs in a dynamic database in which a transaction cannot assume it has locked all objects of a given type. The phantom problem cannot occur if the set of database objects is fixed and only the values of objects can be changed.

Question 2: The source for question two is *Solution for Assignment 4* , cdn.manesht.ir/15572/assgn4Soln.pdf.
2.1
If W(Y) was performed by the transaction T2 before R(Y) is performed by T1, then T2 aborts, T1 would be invalid and T1 would also have to abort.

2.2
T2 would be required to obtain an exclusive lock on Y by the strict 2PL protocol before writing on it. Until T2 commits or aborts this lock would be held. Until T2 is finished, the exclusive lock would block T1 from reading Y and there would be no interference.

2.3
1) Simplicity of implementation: an atomic locking mechanism and a lookup for exclusive locks only need to be provided by the lock manager
2) Ensures only ‘safe’ interleaving of transactions: transactions are recoverable, avoids cascading aborts etc.

3.1 Use READ UNCOMMITTED because inserting a new row in the table Catalog therefore do not need any lock on the existing rows.
3.2 Use READ COMMITTED because updated one existing row in the table Catalog therefore need an exclusive lock on the row which we are updating.
3.3 Use SERIALIZABLE to prevent other transaction from updating or inserting the Catalog table while this transaction is reading from it (the phantom problem).
3.4 Use SERIALIZABLE to prevent other transaction from updating or inserting the Catalog table while this transaction is reading from it (the phantom problem).