1 Is BTO preferrable to 2PL for read-only workloads? Why?
Yes. 2PL requires locking and enforces 2 phases, introducing overhead. In read-only workloads this is not warranted, as all locks can be shared and are hence useless.

2 Would it be a good idea to use BOCC for an OLTP system? Justify!
Nope. Optimistic approaches work well when the expected number of conflicts is low. In an OLTP system there are lots of writing transactions, therefore chances for conflicts are high.

3 Is normal 2PL recoverable? If yes prove it informally or if not show a counterexample.
2PL is not recoverable. When being in the phase of slowly releasing the locks, another transaction can acquire the lock and work on it and commit before the first bigger transaction. If this transaction is aborted, there is no chance undoing the smaller one since it already committed although it is dependent on data of the larger transaction. This could be fixed by using strong or strict 2PL because the write locks are held longer.

4 Compare C2PL to SS2PL. Name one advantage and disadvantage each. How can deadlocks be prevented in both protocols?
C2PL: (+) prevents deadlocks by “default”. (-) usually unfeasible, because operations are not known beforehand. Due to preclaiming, no deadlocks can happen.
SS2PL: (+) produces strict schedules (nice property for recovery). (-) reduces possible concurrency compared to S2PL. Deadlocks can happen, so one can i.e. use wound-wait deadlock prevention to fix this behavior.
5 Consider the following scenario: you need to scan a table with millions of rows and update the updated-at timestamp for each row. This could take a long time. However, you still want to read data at the same time. Which concurrency control algorithm will you adopt? And why?

Altruistic locking. Because the query scans the table, it will not go back to the rows which have already been scanned. Then the lock of a finished record can be donated to another transaction to improve concurrency.

6 Give two examples for deadlock prevention and describe pros and cons for each method.

timeout: (+) can prevent 100% of the deadlocks, simple to implement. (-) hard to set timeout.
wait-die: (+) younger transaction doesn’t need to wait for older transaction. (-) transaction may die several times before getting a needed data item.

7 What is the advantage of FOCC over BOCC?

The flexibility of choosing abort victims or even avoiding some aborts by forcing transactions to wait in their validation phase.

8 What is a desirable property strong 2PL (SS2PL) guarantees? Name one drawback of SS2PL compared to (regular) 2PL?

Gen(SS2PL) ⊆ COCSR, but also Gen(SS2PL) ⊆ Gen(2PL) (i.e. SS2PL guarantees commit order-preserving conflict serializable schedules, but cannot generate as many acceptable schedules as 2PL.)

9 Explain the need for non locking protocols/Mention any two disadvantages of locking protocols.

Locking on a distributed system is hard and synchronization becomes a problem. Locks may themselves end up taking a lot of memory.

10 Consider locking and non-locking schedulers for an OLTP system. Briefly describe in which situations locking schedulers are better than non-locking schedulers and vice versa.

Non-locking schedulers work better in situations where most of the transactions are read-only or read-mostly. Locking schedulers like 2PL will waste a considerable amount of memory and time managing locks. Locking schedulers work better in situations where many conflicts between operations exist. In such a case, non-locking schedulers’ validation phase will fail more often, thus causing an abort of many transactions.
11 Name two advantages of FOCC against BOCC.

A transaction can be easily verified and accepted if it is read-only. This is especially useful if it is known that most of the transactions are read-only. In BOCC a single transaction which writes a value can block/abort all other running transaction which read that value. In FOCC those transaction can commit while the single write transaction has to be restarted.

12 Explain why FOOC is more flexible than BOOC. (Hint: How can a failed validation phase be handled?)

With BOOC the only way to handle a failed validation for $t_i$ is to abort $t_i$. With FOCC there are more options besides the abortion: If the validation of $t_i$ fails, one could abort an active $t_j$ whose read-set intersects with the write-set of $t_i$. Another possibility is to wait and retry the validation again later.

13 Discuss how FOCC can be more flexible than BOCC.

BOCC validates with only the committed transactions, whereas FOCC validates against all parallel transactions which are still in Read phase. This gives FOCC the added flexibility of choosing abort victims, or even holding a transaction until a conflicting transaction has committed.

14 Obviously, SS2PL is the most reliable pessimistic protocol when dealing with locks. Why isn’t it always used?

Concurrency of transactions which access the same resource becomes impossible, therefore reducing the efficiency of the transaction system.

15 What is the difference between locking and non-locking algorithms? Call at least one advantage for each of them.

Non-locking algorithms avoid deadlocks/locking, but instead abort some transactions. Locking algorithms try to let all transactions be executed, but therefore run slower/wait for lock free. Non-locking are faster, whereas locking are more likely to commit all transactions.