You will have 75 minutes for this exam. Good luck! Note: you do not need to fill all the blank space to get full credit; I have purposely tried to give you a lot of extra room.

1. [40 points] Locking.
   
a. [10 points] Draw the compatibility matrix for the locks proposed in the “granularity of locking” paper.

   See the diagram in the Gray et al. paper...

   b. [10 points] Give an example of an anomaly that can happen at the “Snapshot Isolation” level but not at the “Serializable” level, and explain why it can happen at the first but not the second.

   This is “write skew.” For a full description see the “Critique of ANSI Levels...” paper. The idea is that the sequence:

   A5B: r1[x]...r2[y]...w1[y]...w2[x]...(c1 and c2 occur)

   can occur with snapshot isolation, since the writes do not conflict, but not at the serializable level, because the above sequence can allow constraints to be violated that would not be violated by any serial execution.

   c. [10 points] Give the weak and strong interpretation of the unrepeatable read anomaly (that is, give the schedule in the form w1[x]...)

      “Weak” and “Strong” here were confusing, I am sorry for this, the correct terms are “broad” and “strict.” If you gave the correct sequences I accepted whichever label (weak, strong) you used. The schedules are:

      Broad: P2: r1[x]...w2[x]...((c1 or a1) and (c2 or a2) in any order)
      Strict: A2: r1[x]...w2[x]...c2...r1[x]...c1

   d. [10 points] Briefly (only a few sentences!) describe the difference between the use of latches and locks in a DBMS.

      Latches are used to ensure mutual exclusion (only one updater at a time) for data structures that are concurrently accessed by threads/processes in a DBMS. They are lightweight and short-lived.
Locks are used to ensure serializability (consistent execution of multi-step transactions) for data managed by a DBMS that is accessed by multiple transactions. They are typically “heavier” weight than latches, and can be long-lived (they can exist beyond the specific access to the data that they protect.)

2. [30 points] Optimistic Concurrency Control

Give the three conditions (in terms of read sets, write sets, and ordering of read/validate/write phases) that are checked during validation in the parallel validation protocol, and explain how each one is checked.

This is the Kung and Robinson paper. There are three conditions checked by parallel validation. Each condition is a combination of ordering (for example, “Ti completes write phase before Tj starts its write phase”) and properties of intersections of read and write sets.

For full credit, I was looking for answers that correctly stated the three conditions and described where they are checked in the validation algorithm and explains why the validation algorithm actually checks them. In particular, I was looking for answers that said things like “it checks against transactions in the interval <start_tn, finish_tn> because those are exactly the transactions that were active concurrently with the validating transaction that finished their write phase before the validating transaction finished its write phase.” That is, I was looking for answers that explained the mapping from the timing constraints in the conditions to the loops in the validation algorithm.

3. [30 points] B-link tree locking.

a. [15 points] What would go wrong if, upon splitting, we do not insert the new right-sibling in the parent (so that it is only reachable from the left-sibling)? Can you think of an advantage to doing this?

This is a pretty bad idea! The main disadvantage I thought of was that, in the limit, this converts a B-link tree to a linked list, which is horrible for lookup efficiency (linear vs. logarithmic.) It is hard to think of an advantage but the inserts could be quicker if there is no need to update the parent.

b. [15 points] The B-link protocol does not enforce serializability. Why is this? And why is this not a problem?

There were many reasonable answers for “why is this?” I was looking for something like “it uses short write locks and no read locks so it cannot possibly be serializable.”
With respect to “why is this not a problem” I was looking for some discussion of what notion of correctness is needed. Since the B+ tree is not user-level data, the only requirement is that it serves its purpose in the system, which is to provide efficient lookups for searches. As long as it correctly implements those lookups there is no need or benefit to enforcing serializability.