Hints:

- Write your name and student number on the blue book NOW!
- The exam is open book.
- You have 120 minutes for the exam. **Be a smart exam taker:** all points are not born equal. So, if you get stuck on one problem, move on to another. Also, don’t waste your time giving irrelevant (or not requested) details or material. If you say too much and what you say is partially wrong, it may hurt you.
- Clearly identify the question that you are answering for each answer.
- Show all your work. Partial credit is possible for an answer, but only if you show the intermediate steps in obtaining the answer.
- Give answers only in the blue book.

Questions:

1. **[10pts]** In the LogGP model, it turns out that “o” and “g” can affect the performance of parallel applications significantly whereas “L” and “G” have relatively little effects. Explain what L, o, g, G are and why o and g affect performance strongly and L and G do not.

2. **[15pts]** What problems with traditional message passing were the designers of Active Message trying to address? Why isn’t Active Message used more widely today?

3. **[20pts]** Transactions were designed to provide four useful properties: ACID. Explain what each property is and why it is important.

4. **[10pts]** Explain the Chained-Declustering data placement strategy used in Petal. What are the pros and cons of this strategy in comparison to software-based RAID 5 and mirroring?

5. **[15pts]** Would it be easier to build Elephant on top of LFS rather than a normal Unix file system? Why or why not?

6. **[15pts]** Show how a compare-and-swap instruction can be used to correctly prepend an item to a singly-linked list. Would it be easy to implement an append operation with compare-and-swap? Why or why not? Assume the following:

   ```java
   class SLList {
       private head = null; // points to 1st item in list
       void prepend (SLListItem item); // prepend item to list
       // assume that item.next is the // pointer that points to next item.
   }
   ```

7. **[15pts]** Country PPN (Pretty Poor Nation) decides that a bicycle bridge must be built across a deep chasm running down the middle of the country. Unfortunately, PPN can only pay for a one-lane bridge. At most 3 riders can cross the bridge at once if they are going in the same direction. If riders going in opposite directions get on the bridge at the same time, there will be deadlock. Citizens of PPN are unbelievably stubborn and would never consider turning around. Thus, here are the rules that the government has put down:

   There’s a sign visible to everyone saying what direction the bridge can be used in right now. When a rider arrives, if the sign is in the wrong direction but no one is on the bridge, he is allowed to change the sign to his direction and proceed across the bridge. He must wait in line if:
   a. someone is already in line,
   b. the sign has the wrong direction and someone is on the bridge, or
   c. the sign has the right direction but there’s already 3 riders on the bridge, or
When a rider successfully crosses the bridge, if someone is waiting in line on the other side, he sets the sign to the other direction. If he is the last one off of the bridge, he can tell up to 3 people in line on the other side to use the bridge. On the other hand, if no one is waiting, he can shout back across the bridge, telling someone in line that he can now use the bridge.

Each rider calls the following routine, which must implement the above set of rules correctly. Write pseudocode for this routine using locks, condition variables, and/or semaphores.

```plaintext
cross_bridge(direction) {
    // returns when have successfully cross bridge
}
```