**15-440 (Fall 2008)**
**Homework #3**

1. In designing a solution using Map-Reduce, what is generally preferable, a greater or fewer number of phases? Why?
2. Why does the fabric sort the records before performing a Reducer? Are they also sorted prior to processing by a Combiner?
3. Can you change the way the records are sorted? If so, how? If not, why not?
4. Is it possible to cause records to be sorted by value, rather than by key, before Reducing? If so, how? If not, why doesn't this make sense?
5. Asymptotically, how much memory is too much for use in a Mapper? A Reducer? Why?
6. When is it desirable to use a Combiner?
7. What is the difference in the way Combiners and Reducers get their input and output? Why the difference?
8. Does the system recover if a Mapper node dies? If so, how? If not, why is this a negligible concern?
9. Does the system recover if a Master node dies? If so, how? If not, why is this a negligible concern?
10. How does one get access to the results of a Map-Reduce program?
11. How does the data initially get to the Mapper nodes? Why aren't bandwidth and latency big concerns?
12. How does the data get from a Mapper node to a Reducer node? Are bandwidth and latency significant concerns?
13. What happens if a Reducer is asked to scarf up more records than it can fit into memory? Are they sorted? If so, how?
14. What is the role of the Partitioner?
15. If you write a Partitioner for use in place of the default, what aspects of a record may be considered by the Partitoner?
16. What factors determine the optimal number of Mapper nodes? Reducer nodes?