Q Exam ID#______

Systems Q Exam – Spring 1999

Please check to see that you have 10 pages and 5 questions. Please put your Q ID number on each of the pages. You have 2 hours to complete the exam.

1. (10 points)
Decision-support applications may consume large amounts of data. For instance, Wegmans has a multi-gigabyte database server representing all its sales. Wegmans executives usually interact with the system through a web browser running on a desktop computer.

a) Sketch a realistic end-to-end architecture for the data analysis system (i.e. draw a block diagram with a block for the database server, a block for the browser, any other blocks in between, and labeled lines between them indicating flow of data and control).

b) Suppose main-memory suddenly became so cheap that the entire Wegmans database could be easily stored in main memory. The Wegmans IT department needs to choose between retaining their existing relational database server or buying a new "main-memory database server" instead. List two ways in which such a new server might differ from a traditional RDBMS, and two ways in which it is similar.

2. (12 points)
A system supports an unlimited number of threads labeled t0, t1, ... Threads may participate in a special class of actions called "zap" actions. Zap actions form a partial ordering. If t_i and t_j are both trying to execute a zap action simultaneously, $t_i < t_j$ means that thread i zaps before t_j, $t_j < t_i$ means thread j zaps before thread i, and t_i and t_j are unordered if they can zap concurrently. Assume that the thread identifiers (such as i and j in the examples above) are 64-bit integers named "id". Thus, given a pointer 'tp' to a thread one might write 'tp->id' to reference its id.

a) Propose efficient data structures and algorithms for representing the partial order on zap actions. You can associate additional per-thread information with each thread and can use a high-level language notation which supports sets and set operations in the usual manner.

b) Give code for a procedure before_zap (thread *tp) and a procedure after_zap (thread *tp) which enforces the partial order. For example, if $t_i < t_j$, then t_j will wait in before_zap until t_i has called after_zap. Assume that each thread calls before_zap, zap, and after_zap exactly once.
Note: "Representing" a partial order on threads means that you should describe a data structure capable of supporting the "<" operator. You should tell us how you want this data structure to look, but don't need to write the code to initialize it or to implement "<". Having done this, your code for begin_zap and after_zap can make use of the "<" operator on thread pointers, for example in a statement like "if $t_i < t_j$....", where t_i and t_j are pointers to threads, or (if you prefer) thread identifiers. You may find it helpful to think of this as an overloading of the "<" operator for a "thread" abstract data type. (However, if you have some other idea for solving the problem, there is no requirement to define or use the "<" operator).

3. (10 points)
Most of the work done by a typical Web server is in reading files from a disk and writing them to the network. Consider a page that is read from the file system and written to an Ethernet host adapter card.

a). How many times is the page copied in a popular operating system of your choice (Unix, NT, Be etc.)? List each time a copy is made and state why it is necessary.

b) Describe two nontrivial techniques to reduce the number of copies and explain the advantages and disadvantages of each technique. Hint: you may consider modifications to the OS, the network protocol stack, or the web server.

4. (12 points)
Programming languages, such as Java, that support monitors generally have some form of "wait" and "signal" statements for synchronizing processes in a monitor. For condition variable c, a process executing c.wait relinquishes exclusive access to the monitor and becomes blocked on c. There are several choices for the semantics of c.signal. In all cases, execution of c.signal causes one or more processes previously blocked on condition variable c to again become eligible for execution. Why might each of the following choices for additional semantics of c.signal be the right one?

(a) c.signal must be the last statement in a monitor procedure

(b) c.signal causes its invoker to become suspended, some waiting process reactivated, and only when mutual exclusion of the monitor is relinquished is the signaler allowed to continue executing.

(c) c.signal causes all processes blocked on condition variable c to be marked as eligible for execution in the monitor.

(d) c.signal does not suspend its invoker.

5. (10 points)
Processors are using ever deeper pipelines to improve performance. This is because shorter pipe stages have lower propagation delays and can thus be clocked at higher frequencies. Give two reasons for which one can't simply keep slicing the pipeline into more stages to increase overall processor performance.