ECE459 Midterm W2014

Question 1

a) Not covered. Adding restrict tells the compiler that for the lifetime of the pointer, no other pointer will be used to access the object to which it points.
b) Not covered.
    a. int x = 3
       int x = 4
    b. int y = 3
       int x = 4
c) Since there are more threads than cores in the 9-thread version, there is added overhead from context-switching between the threads, since only 8 of them can run at a time. There is also potentially additional resource contention.
d) Not covered.
e) Not covered.
f) It allows variation of the problem size (e.g. input size, number of timesteps, etc).
g) The threads continue to consume resources or memory, which can lead to increased resource contention or resource exhaustion.
h) Non-blocking I/O would be better for this scenario. This would allow us to effectively manage only the active connections, without wasting resources on inactive connections. This also eliminates the overhead of thread startup/shutdown and context switching for 300,000 threads.
i) Data parallelism. Multiple threads perform the same operation on separate data items.
j) Not covered.

Question 2

a) The lock is created and redefined every time find_and_increment is called. To fix this, we can initialize the lock upon program initialization, outside the find_and_increment function.

b) We first remove global_lock. Then, we add a new struct data { int val; pthread_mutex_t lock }. We change the type of data in node to be data* data, and add a field pthread_mutex_t lock. Now, before modifying data on line 18, we first lock n->lock, then we also lock n->data->lock. After updating n->data->value, we unlock n->data->lock and then unlock n->lock. In other words, we have a lock protecting the data pointer, as well as a lock protecting the value of the data.

Question 3

Not covered.

Question 4

Not covered.