The Final Exam will be Wednesday, Dec. 12 from 1 PM to 2:50 PM in our normal classroom (ITT 322). It will focus on the material since the mid-term, but also cover the main ideas from chapters 1 through 5. It will be open-book and notes. The material since the last test includes the following topics (and maybe more).

Chapter 6. Programming with Threads
General thread concept
POSIX Threads operations: pthread_create, pthread_join, pthread_exit, pthread_self
Thread attributes
Mutual Exclusion: pthread_mutex_lock, pthread_mutex_unlock, pthread_mutex_trylock,
Dynamic creation and destruction of mutexes
Thread synchronization: pthread_cond_wait, pthread_cond_signal, barrier implementation
Dynamic creation and destruction of condition variables
Thread-Specific data
Deadlock, lock hierarchies, monitors
Performance Issues: granularity, thread scheduling, priority inversion
Case Study: 2-D Successive Over-Relaxation (SOR) using pthreads - barrier implementation, split-phase barrier idea, corrected code at: http://www.cs.uni.edu/~fienup/cs6400f12/lectures/fig6-16C2.c

Chapter 7. MPI and Other Local View Languages
General concept of MPI and basic MPI functions (e.g., MPI_Init, MPI_Comm_size, etc.)
Corrected Count 3s MPI code at: http://www.cs.uni.edu/~fienup/cs6400f12/lectures/fig7-1.c
MPI Send, Recv, Reduce, and Scatter functions, Groups and Communicators, point-to-point communication
Other communication modes
Collective Communication: Scan, Bcast, Barrier
MPI example: 2-D Successive Over-Relaxation (SOR)
Derived data types

CUDA Programming: NVIDIA slides at:
http://www.nvidia.com/content/cudazone/download/Getting_Started_w_CUDA_Training_NVISION08.pdf
UW - slides at: http://sbel.wisc.edu/Courses/ME964/2012/Lectures/cudaNegrutWisconsin.pdf
General concept of CUDA architecture:
- heterogeneous system (host CPU and GPU device) with separate memories,
- host manages (allocates, copies between, and frees) both memories
- host launches a grid (1d or 2d) of thread blocks (each block can be 1d, 2d, or 3d),
- 1000's of lightweight threads run concurrently SIMD and communicate via a shared-memory model
- lightweight thread synchronization primitives (`__syncthreads()` within a block)

Types of device memories:
- global memories: global memory, constant memory, texture memory (off-chip)
- shared memory (on-chip)

Built-in variables for each thread to identify itself:
- threadIdx (uint3: .x, .y, .z) - thread index within a block
- blockDim (dim3: .x, .y, .z) - dimension of the block
- blockIdx (uint3: .x, .y) - block index within the grid of blocks
- gridDim (dim3: .x, .y) - dimension of the grid
- warpSize (uint) - SIMD chuck of threads within a block

Atomic Operations on integers in global memory (compute capability 1.1+) or in shared memory (compute capability 1.2+)
CUDA Examples: Count 3s, Matrix Multiplication, 2D SOR, TSP Tree search (buggy)