Homework 4, due Monday, December 5 at 11:59PM

Instructions: We'll go over these in class on December 1. Handin on CADE machines:

“handin cs4961 hw4 <proffile>”

1. Given the following sequential code, sketch out two CUDA implementations for just the kernel computation (not the host code). Determine the memory access patterns and whether you will need synchronization. Your two versions should (a) use only global memory; and, (b) use both global and shared memory. Keep in mind capacity limits for shared memory and limits on the number of threads per block.

```c
int a[1024][1024], b[1024];
for (i=0; i<1020; i++) {
    for (j=0; j<1024-i; j++) {
        b[i+j] += a[j][i] + a[j][i+1] + a[j][i+2] + a[j][i+3];
    }
}
```

2. Programming Assignment 3.8, p. 148. Parallel merge sort starts with \( n/\text{comm}\_sz \) keys assigned to each process. It ends with all the keys stored on process 0 in sorted order. ... when a process receives another process' keys, it merges the new keys into its already sorted list of keys. ... parallel mergesort ... Then the processes should use tree-structured communication to merge the global list onto process 0, which prints the result.

3. Exercise 6.27, p. 350. If there are many processes and many redistributions of work in the dynamic MPI implementation of the TSP solver, process 0 could become a bottleneck for energy returns. Explain how one could use a spanning tree of processes in which a child sends energy to its parent rather than process 0.

4. Exercise 6.30, p. 350 Determine which of the APIs is preferable for the n-body solvers and solving TSP.
   a. How much memory is required... will data fit into the memory ...
   b. How much communication is required by each of the parallel algorithms (consider remote memory accesses and coherence as communication)
   c. Can the serial programs be easily parallelized by the use of OpenMP directives? Do they need synchronization constructs such as condition variables or read-write locks?