1. **(20 points)** Give a one or two sentence definition for each term or phrase below (5 points each).

   (a) False sharing.

   (b) Simultaneous multi-threading.

   (c) One-sided communication.

   (d) Tail recursion.

2. **(40 points)** The following questions pertain to the paper, *MapReduce: Simplified Data Processing on Large Clusters*. Each of these questions has a value of 10 points.

   (a) The paper describes a Map-Reduce paradigm. Would it be practical to extend this to Scan-Reduce? Why or why not?

   (b) Briefly describe what happens if a machine taking part in a Map-Reduce computation crashes before the computation is complete. Describe a Map-Reduce computation where the result is the same whether or not some machine crashes. Describe a computation where a different result can be produced because a machine crashes. Your answer to all three of these questions about machine crashes should take *at most* eight sentences.

(c) What would be a reasonable value for λ (from the CTA model) for Map-Reduce computations performed on a large cluster? Given a estimate and justify your answer with one or two sentences.

(d) Here's a common data mining problem. Let $\mathcal{S}$ be a set of sets. For any pair of elements, x, and y, let

$$\text{togetherness}(\mathcal{S}, x, y) \quad = \quad |\{R \in \mathcal{S} \mid (x \in R) \text{ and } (y \in R)\}$$

In English, $\text{togetherness}(\mathcal{S}, x, y)$ is the number of sets in S that contain both x and y. For example, each subset of S could be the set of items purchased in a single transaction at Amazon. In this case, a high value for $\text{togetherness}(\mathcal{S}, x, y)$ would mean that Amazon would tell you that x and y are frequently purchased together.

Write pseudo-code for map and reduce functions that could be used to compute togetherness. You can use the example code from the top of page 108 of the paper as an example for syntax, calling conventions, etc.

3. **(30 points)** The following questions pertain to the paper, *The GPU Computing Era*. Each of these questions has a value of 10 points.

   (a) Assume that each CUDA core can perform one double-precision, floating-point operation every 4ns. What is the peak-computation rate for the Fermi GPU shown in Figure 3?

   (b) A double-precision floating point number is eight bytes. Assume that each double-precision operation has two operands. If each value stored on a Fermi GPUs caches and registers is used once, how long does it take for the GPU to consume all of its on chip data?

   (c) A Fermi GPU can read data from off-chip DRAM at a peak rate of 144GB/sec. If a Fermi GPU is to operate at its peak-computation rate, how many double-precision operations must it perform for each double-precision value read from memory? For simplicity, you can pretend that there is nothing to be written back to memory.

4. **(10 points)** Do any one (but not more) of the following three questions:

   (a) Sketch a way to compute matrix-multiplication using Map-Reduce.

   (b) Sketch a way to compute count-3s using CUDA.

   (c) Given an array $x[i]$ for $0 \leq i < n$, use generalized reduce or scan to compute $y[i]$ with

$$y[i] \quad = \quad x[0]^{x[1]^{x[2]^{\cdot^{\cdot^{\cdot^{x[i]}}}}}}$$