1. [2] If there are too many threads that want to use a lock, that lock becomes heavily contended and a big source of performance loss. If you are faced with such a situation, what will you do?

2. [3] If three threads, in OpenMP, execute the instruction x++ where x is a shared variable initialized to 0, what are the possible values that x could have after the execution of the threads (assume no synchronization or precautions were taken)?

3. [2] In bulleted list state the source(s) performance loss that we may face in OpenMP.

4. [3] Assume we are multiplying an 8000x8000 matrix with vector y. Each element of the matrix and vector is double float (i.e. 8 bytes). Also suppose that thread 0 and thread 2 are assigned to two different cores. If a cache line contains 64 bytes. Is it possible for false sharing to occur at any time between threads 0 and 2?

5. Assume we have a dual-core processor and we are doing a matrix vector multiplication where the matrix is 8x8,000,000 and we multiply it by vector y. Each element is a double float (i.e. 8 bytes) and a cache line is 64 bytes. We parallelized this program using 4 threads.
   a. [2] What is the minimum number of cache lines that are needed to store vector y?
   b. [2] What is the maximum number of cache lines that are needed to store vector y?

6. [3] How many threads could be used for the computation below, each thread can execute one or more of the instructions:
   x++;
   a = x + 2;
   b = a + 3;
   c++;

   without changing the code. Explain clearly your answer.

7. [2] According to Amdahl’s law, what is the maximum speed-up of a parallel computation given that 80% of the computation can be executed in parallel?

8. [1] Can a serial program be faster than a parallel version? Explain?