**CSCI-GA.3033-016**
**Multicore Processors: Architecture & Programming**
Homework Assignment 1

1. [3] There are several types of parallelism that we can find in different programs. What are they? For each one, specify whether exploiting that type needs programmer involvement or the hardware/compiler is enough to exploit it.

2. Multiprocessor systems, where we have several chips each of each is a single core, have been around for several decades now. This means we should already have good experience dealing with parallel systems. Yet, we are facing challenges dealing with multicore processors.

   a) [2] List all the differences you can think of between traditional multiprocessor systems and the current multicore processors.
   b) [1] List one or more cases where our expertise with traditional multiprocessor systems is helpful in dealing with multicore processors.
   c) [1] List one or more cases where our expertise with traditional multiprocessor systems is NOT helpful in dealing with multicore processors.

3. [3] What do you think are the factors that can make an application very hard (or sometimes impossible) to parallelize?

4. [1] If you are given a sequential program that you are required to parallelize, first you need to find the parts that are *parallelizable*. However, in some cases, it is not worth it to parallelize those parts, why?

5. [2] Suppose you have two parallel programs that solve the same problem. State two factors that can make you pick one program over the other (beside price of course).

6. [2] Suppose, for a specific problem, we know the best algorithm for it for a single core (e.g. quicksort is best for sorting). Does this mean that this algorithm is also the best for multicore? Justify.

7. Suppose we have the algorithm (assume N is a large even number):

```
for(i = 0; i < N/2; i++)
        a[i] += a[i+ N/2 ];
```

   a. [3] Can we parallelize the above algorithm? If no, why not? If yes, explain.
   b. [2] What is the maximum number of cores after which no performance enhancement can be seen? Justify