**Programming Assignment 4**

**50 points**

Your task is to design and develop a parallel solution to the Floyd-Warshall algorithm. You hopefully covered Floyd-Warshall in your algorithms course.

Your goal is to design and develop a multithreaded approach to this problem, and to substantially improve the performance of this algorithm when running on a multiprocessor system. You should follow the steps covered in the Parallel Programming Design course handout: partitioning, communication, agglomeration and mapping.

With your program, you should submit a design document, explaining your design decisions for each part of the design process. Your implementation should correspond to your design. In your design document, include an introduction, and the followings sections. Your document must concisely cover all the content, but approximately 3 pages, double-spaced in 12-point font would be a good estimate. Concise prose documenting your design is sufficient.

**Partitioning**

First, choose at least two ways to partition the problem. As we are finding the volume of an area that falls on a rectangular plane, several possible partition designs should be apparent. Ensure that each design has many more partitions than processors. You should describe specifically what portions of each program fall into each partition. Indicate whether you are performing functional or domain decomposition.

**Communication**

After subdividing the problem, different sections must communicate in some way to calculate total results. Design how to communicate between partitions, for each partition you chose. You must describe, for each partition scheme you chose, what communication is required to calculate the result.

**Agglomeration**

For each partition, consider how to combine your initial partitions to reduce partitioning and overhead.

**Mapping**

Assign tasks to threads to minimize communication costs, and to maximize parallelism.

**Implementing and testing your program:**

You are provided a single-threaded version of the algorithm. You may choose to modify the existing code, or develop your own. Your program should use the System.nanoTime() method to print the amount of time each run takes. You should run your program on a matrix of size 5000 \* 5000, as randomly generated in the example code. Run your program for thread counts of 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10 threads. Report how many CPUs your computer has. If possible, run your code on a computer with at least four CPUs.

You may use any multithreading constructs you wish, but I hope that your program will produce results with a speedup (per Amdahl's law) of at least 1.8 on two CPUs, 2.7 on three CPUs, and 3.6 on four CPUs.