CSCI 5646 **Problem Set 1**

This assignment is due by 4pm on Friday, September 2nd. Minimal credit will be given for incomplete solutions or solutions that do not provide details on how the solution is found. You are encouraged to discuss the problems with your classmates, but all work (analysis and code) must be your own.

1. Let $\mathbf{x}$ and $\mathbf{y} \in \mathbb{R}^n$ and consider computing integer powers of their outer-product, i.e. $B = \left(\mathbf{x}\mathbf{y}^T\right)^k$. A naive implementation (i.e. forming the outer-product matrix and then taking powers) would require $\mathcal{O}\left(kn^3\right)$ FLOPS. However, a clever implementation can be found that runs in $\mathcal{O}\left(n^2\right)$ FLOPS. Write pseudocode for such an algorithm in the spirit of Algorithm 8.1 from the textbook. Analyze the complexity and give the leading order term (including any relevant constants).

2. Implement functions `axrow` and `axcol` which perform a matrix-vector product in row-oriented and column-oriented fashions, respectively. Time your functions on square matrices of size $n = 2^k$ for $k = 1, 2, \ldots, 12$ and plot n vs. runtime for both functions on the same set of axes (feel free to go to larger matrices to better see the separation). Based on your results, which matrix storage scheme does your chosen language use? Turn in a printout of your mat-vec functions as well as any driver function you used. (**Note**: See the associated post on Piazza for implementation and timing tips in Matlab and Julia.)

3. Implement functions that perform a matrix-matrix product in the style of (1.1.13) for each of the six possible orderings of the for-loops. Time your functions on square matrices of size $n = 2^k$ for $k = 1, 2, \ldots, 10$ and plot n vs. runtime for all six functions on the same set of axes. Your results should divide the orderings into three pairs from fast to slow to really slow. Based on the matrix storage scheme that you identified in Problem 2, give a plausible explanation for the relative speed of the orderings based on data movement in and out of cache. For this problem you don't need to submit any code, just turn in your plot and your explanation.