Matrix multiply and graph search
ECE 563 Spring 2018 Small Course Project
version 1.0 (1/17/1018)

This project involves implementing a matrix multiply and a binary tree search using OpenMP and MPI. For both the matrix multiply and the binary tree search your code should be *self-initializing*, that is, you don't need to read any files but can simply internally initialize arrays to be multiplied and the tree to be searched.

Step 1: Perform a matrix multiply and tree search using OpenMP. Show the speedup for 2 and/or 4 cores, depending on how many cores the machine you have contains.

Step 2: Perform a matrix multiply and tree search using MPI. Show the speedup for 2, 4, 8 and 16 nodes.

Step 3: Perform a matrix multiply and tree search using the results of our discussion of how to do these later in class. Find the speedups for 2, 4, 8 and 16 nodes for both implementations. perform a Karp-Flatt analysis and an iso-efficiency analysis on the MPI version.

Note: While the problem description says the tree is a binary tree, it does not say that nodes are inserted in the order you will be searching. This is important. For example, consider a binary tree where nodes are of the form <*key, data, left, right*>. Let the *key* be the name of a person and the *data* be their age. Thus the tree will be arranged by name but you might be asked to find the names of everyone who is 25 years old. In this case, the entire tree would need to be searched.

Note: It is perfectly ok to just have a tree whose nodes are numbers, but leave them unsorted.