Homework #3
Due: Thursday, Feb 1, 2018 by Midnight ET

CS 1645/2045
Spring 2017

Question 1 of 2:

In class we discussed matrix vector multiplication (slides 22-25, book Section 4.3). In this question you should implement matrix vector multiplication using pthreads. Most (but not all) of this code is already provided either in my slides or in the book.

a. To implement this algorithm start with the boilerplate code found in the github repository, you should be able to "`git pull`" in the same directory you created in HW2 (on the comet system). Or you can create another source tree by doing this:
   `git clone https://github.com/bryanmills/hpc-course-2017.git`
b. Once you have hw3 directory.
   ```
   cd hw3/matrixvec
   Make
   sbatch matrixvec.batch
   ```
   You can monitor this job using:
   `squeue -u <your comet username>`
c. Once completed you will have matrixvec.o<jobid> in your directory. Look at its contents, note that the matrix-vector is not actually multiplied.
d. The program provided does not implement any of the actual functions, it just prints out the matrix and vector. Your job is to implement the method:
   `nth_mat_vector(void* rank)`
e. After this method is implemented you then need to create a new thread NUM_THREADS times and call the method you just implemented. You'll see I already created this loop for you in the main() function, you need to use the pthread_create method to call your newly written method.
f. You now need to join the threads back together. This forces the main thread to wait on all the other threads to complete before printing out the final result. Use pthread_join to complete this task.
g. Once you believe this is correctly implemented you can re-compile and run the program again using the squeue command. Verify the output in the file and that the math is correctly implemented.
h. The code provided in the lecture notes and in the book assumed that the matrix and vector were evenly divisible into NUM_THREADS. Using the example code, modify the NUM_THREADS to be 3. Does your implementation still work? If not modify your implementation to handle this case.

Question 2 of 2:

Write a parallel program using Pthreads that implements Strassen's algorithm for matrix multiplication. The algorithm computes the multiplication of N × N square matrices C = AB by splitting the matrices into a series of 2x2 matrices:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix}$$

These matrices are constructed by splitting the original matrices into equally sized block matrices. If the original matrices do not evenly fit into 2^n x 2^n we can pad the matrices with zeros. The magic sauce of Strassen's algorithm is to calculate the following intermediate matrices.

$$\mathbf{M}_1 := (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2})$$
$$\mathbf{M}_2 := (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\mathbf{B}_{1,1}$$
$$\mathbf{M}_3 := \mathbf{A}_{1,1}(\mathbf{B}_{1,2} - \mathbf{B}_{2,2})$$
$$\mathbf{M}_4 := \mathbf{A}_{2,2}(\mathbf{B}_{2,1} - \mathbf{B}_{1,1})$$
$$\mathbf{M}_5 := (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\mathbf{B}_{2,2}$$
$$\mathbf{M}_6 := (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2})$$
$$\mathbf{M}_7 := (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2})$$

Then, calculate the subparts of C.

$$\mathbf{C}_{1,1} = \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7$$
$$\mathbf{C}_{1,2} = \mathbf{M}_3 + \mathbf{M}_5$$
$$\mathbf{C}_{2,1} = \mathbf{M}_2 + \mathbf{M}_4$$
$$\mathbf{C}_{2,2} = \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6$$

Note that the calculations of the M_i and $C_{i,j}$ matrices are all independent, meaning they can be parallelized.

a. To implement this algorithm start with the boilerplate code found in the github repository
b. Once you have hw3 directory.

```
cd hw3/strassen
make
sbatch strassen.batch
```

c. There is an starter file (implemented serially) found in the git repository located here: https://github.com/bryanmills/hpc-course-2017/tree/master/hw3/strassen
d. This should then queue a 16x16 matrix multiplication using the simple method. Your job is to implement Strassen's algorithm in this file, there are further instructions in the C file.
e. Start by implementing a sequential function that implements the algorithm.
    i. Pad matrix if necessary, define matrixes (M_i, $C_{i,j}$), Do the math.
f. Then determine how best to parallelize the steps being taken in the serial version.