CX 4010 / CSE 6010 Assignment 1: Matrix Multiply

Due Dates:

- Due: 11:59 PM, Thursday, September 4, 2014
- Revision (optional): 11:59 PM, Monday September 8, 2014

The objectives of this assignment are to write a program to compute the product of two dense matrices, C = A × B, and to understand the computational performance of your program.

1. Write a function with the prototype

```
int matrix_multiply(int n1, int n2, int n3,
double *a, double *b, double *c);
```

   The matrix A is stored in array a and has dimensions n1 by n2, and the matrix B is stored in array b and has dimensions n2 by n3. When your function returns, the product C = A × B is stored in array c. Your function should return 0 if the computation was successful, and nonzero otherwise (e.g., n1 is negative, a==NULL, or any other invalid condition). From the function prototype above, note that memory for the arrays must be allocated outside the `matrix_multiply` function.

2. Write a function with the prototype

```
int matrix_fill_random(int n1, int n2, double *a);
```

   that fills the matrix A with random values between −10 and +10. The return value of the function specifies whether or not an error occurred.

3. Write a function with the prototype

```
int matrix_print(int n1, int n2, double *a);
```

   that prints the matrix A in a readable format (i.e., columns are aligned).

4. Write a program called matrix test whose main function calls the above three functions. Use this program to test your functions. In particular, choose A as a 4 × 3 matrix, and B as a 3 × 2 matrix. For this case, print A, B, C and hand in your results. Also hand in evidence that you checked that C is correct, e.g., a Matlab script that shows the same computation and results.

5. Write a program called matrix benchmark. This program prints the average time for computing one matrix multiply. The pseudocode for this program looks like this:

```
matrix_fill_random(A)
matrix_fill_random(B)
start timer
loop a large number of times
        C = A*B
```

endloop
stop timer
print average time for one matrix multiply

The reason for the loop above is to average out any startup effects (i.e., this "warms up" the caches).

6. Run matrix benchmark using square matrices where the dimension of the matrices ranges from 100 to 1000, or as large as your machine can handle. Plot the average time as a function of the cube of the dimension. Provide an explanation of your results. (You may find it useful for your program to input the matrix dimension from the command line.)

Turn in a brief report including your results, and your software in a single zip file. Your software must be well documented and include comments so the code is easy to understand. You should include a README file with instructions on how to compile and run your program on the jinx cluster. Although this is not really necessary due to the relatively simple nature of this program, you should get into the habit of including such documentation with the software you develop. Graduate and undergraduate students will complete the same project, but will be graded on different scales. Finally, please note that all code you turn in must be completely developed by yourself, although we encourage you to discuss the problem and issues you are facing with other students (as well as the TA/instructor).