CS360 – Spring 2015
Project 1: Matrix Multiplication

First carefully read the general instructions that apply to all the projects in this course. Next, using either Java 6 or C++2011, write a program that solves the following problem.

Input modes:

1. Obtain a single integer parameter *n* from the command line. Validate that the argument is an integer. You will initialize the two input *nxn* matrices with random values from 0 to 9.

   ```
   %. /project1 –n 6
   ```

2. Read a file to obtain the two input matrices. You can assume that the file is formatted correctly (numerical values that are comma-delimited and the matrices are square, etc). The file name will be datafile. Below is an example of the data file. The first line contains the n dimension. There will be n*2 lines following the initial line that denotes the n value. The values in the matrices can vary from 0 to 99.

```
n=6
1,2,3,4,5,6
1,2,3,4,5,6
1,2,3,4,5,6
1,2,3,4,5,6
1,2,3,4,5,6
1,2,3,4,5,6
1,2,3,4,5,6
1,2,3,4,5,6
1,2,3,4,5,6
1,2,3,4,5,6
1,2,3,4,5,6
1,2,3,4,5,6
```

Functionality

Either generate the random matrices or read the matrices based on the presence of the –n option on the command line. If the command line argument is provided, generate the matrices randomly according to the rules given above. If no command line argument is given, the matrices should be read in from the data file. Finish creating the matrices before beginning the multiply operations.

Calculate the product matrix using Strassen's method. Also accumulate a count of the number of multiplications and additions that are performed. You can complete multiple operations per statement. Just increment your counters by the appropriate value.

Use a two-dimensional array of integer primitive types (make sure you know what this means in Java). You will be able to allocate them based on the dimension provided in the file or on the command line. You only need to read the file once. Make sure that you parse the input lines as efficiently as possible. Use the built in functions where it makes sense.

Output should contain the 1) value of n, 2) formatted input, 3) formatted output, and 4) the count of the multiplication and addition operations. The format is given below:

```
N=6

Input matrix A
    1     2     3     4     5     6
    1     2     3     4     5     6
   22    22    22    22    22    22
    1     2     3     4     5     6
   11    11    11    11    11    11
   22    22    22    22    22    22

Input matrix B
    1     2     3     4     5     6
   31    31    31    31    31    31
   42    42    42    42    42    42
    1     2     3     4     5     6
   11    11    11    11    11    11
   22    22    22    22    22    22

Output matrix C

    380    385    390    395    400    405
    380    385    390    395    400    405
   2376   2420   2464   2508   2552   2596
    380    385    390    395    400    405
   1188   1210   1232   1254   1276   1298
   2376   2420   2464   2508   2552   2596

Number of multiplications: XXX

Number of additions: XXX
```

Failure to follow directions will result in point deductions or a zero. There are 83 students in this class. It is unreasonable to expect that any exceptions to the procedure will be made.

Late assignments will not be accepted unless you have a doctor's not covering the entire period from Jan 14- Jan 25. Java programs should not require any machine specific code. The C++ programs will be tested on bama.ua.edu. You can gain access to this system for testing. Develop C++ on other platforms at your own risk. Programs must only include constructs made available by the language specifications (Java 6 and C++11/Posix- https://en.wikipedia.org/wiki/C%2B%2B11) and no other extensions. You also must comply with any prohibitions on data structures in the program description.

Blackboard submission by Jan 25th 11:59pm:

- Program code in ASCII file with .java or .cpp extension. Pdfs of code will not be graded.
- Test results in pdf format. You should generate test data such that it shows that your program works.
- Unix batch (bash or csh), make or ant file to compile and run the test cases. File should run without any machine specific requirements other than make, bash/csh and ant.
- Any relevant notes in text file README

In class printout submission Jan 26th @ 9am:

- Program printout that is easy to read (line wrapping etc).
- Short version of test results (save a tree…)

- The source code (.java or .cpp files) should also be turned in via Blackboard (not emailed to me or the TA). I do not use a Windows computer. Test result submissions of any type other than pdf will not be graded.

- If you forget to turn in your test results, your program may be graded as if it does not work.

- The source code and test results should be printed and brought to class on Jan 26th at 9am to get credit. Any assignments that are not submitted via printout on Jan 26th will receive a 50% deduction.

- You can check you answers here http://octave-online.net. It uses the matrix input format of Matlab.

- I will do some initial automated testing. If you name the file something else, I have to go into your code, fix the problem and recompile or rename the file to match your file name.

- This is an individual assignment. The program must represent your own work. You can discuss high-level concepts. Do not show your code to anyone. I reserve the right to ask you about your program to discern if you indeed wrote the code. If you cannot explain your code and choices verbally, you may be turned in for academic misconduct. All submissions will be analyzed to identify possible cases of cheating. Any cases of suspected collaboration

will be referred to the College of Engineering Dean. A zero or low grade is always better than having an academic misconduct on your academic record.

Here is an overview of evaluation criteria. It is subject to change within 10% at my discretion. The top submissions that contain particularly elegant and clear code may earn extra credit.
70% - Functionality

- Program contains the correct code to produce the output.
- Program meets the requirements (correct multiplication, data structure, etc)
- Only hardcoding includes the data file name
- No artificial limits on the matrix size (of course computation will limit us)
- Output from running the program is included that shows that the program works
- Program uses appropriate language constructs
- Overall design should be reasonably consistent with OO design principles

30% - Design

- Program exhibits defensible design choices in algorithms and data structures (if you add any)
- Program does not contain extra loops or any code that hurts efficiency
- Program must use appropriate and consistent style for naming of elements
- Program must include reasonable whitespace and appropriate indentation
- Program must include comments, especially in areas where you need to support your choices or where the purpose of the code is unclear.

** Clarifications on the assignment will be posted to blackboard. You will be responsible for consulting blackboard frequently and remaining aware of clarifications.