CS575: Programming Project 2
Due: 11:59pm on March 16

Instructor: KD Kang

1. [50%] **Strassen's matrix multiplication algorithm.** Your program should take an input variable $n > 0$ in the Linux command line, generate two n*n random float matrices, A and B, and compute A * B using the Strassen's matrix multiplication algorithm for the following two cases. To generate A and B, generate random float numbers that range between -5.0 and 5.0. For random float numbers consider two digits after decimal points (like 3.45)
   (a) [25%] $n = 2^k$ where k is a positive integer.
   (b) [25%] $n \neq 2^k$.

   In addition, implement the standard matrix multiplication algorithm with $O(n^3)$ time complexity. Compute A*B using Strassen's algorithm and compare the result to the result produced by the standard matrix multiplication algorithm. Print both results for comparisons. (If incorrect results are produced, no credit will be given. Your program should work for any matrices. If it works for specific matrices but doesn't work for other matrices, no credit will be given.)

   Your c or cpp file for strassen's algorithm should contain two basic functions. **strassensMultiplication()** and **standardMultiplication()**. You main() function should be able to read inputs and generate two matrices. If you need additional functions to process random matrices, you can create your own. Make sure both strassensMultiplication() and standardMultiplication() functions prints the result before the return statement. These two functions will be called from main() function. Make sure you put appropriate one or two line comments to each of your function definitions. Inappropriate function declaration and unnecessary function calls will cost your points.

   if your multiplication function is recursive then use strassensMultiplication() as a wrapper function and call your own custom recursive function from it. Do similar for standard function too. DO NOT print unnecessary result other that mentioned below sample output.

   Example:
   ```
   void  strassensMultiplication(parameters)
   {
       customRecusriveStrassensMultFunction();
       print result here;
      return;
   }
   ```

   For strassen's matrix multiplication, the c or cpp file name should be **strassen.c or strassen.cpp.** Your makefile will generate .out file as strassen.out. Make sure your makefile will not run/execute the executable file.

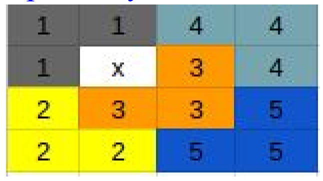
   Your command to run the Strassen program should be: `./strassen.out <value of n>`

   The TA will run this program for both the above mentioned conditions and will test the output. For each case your program will generate matrices A and B, the output of Strassen's algorithm and output of standard multiplication algorithm. Don't print unnecessary texts or debug comments.

Your output will look like below (don't go with the exact multiplication value, this is just an example):
Matrix A:
```
0   0   0   0
0   0   0   0
0   0   0   0
0   0   0   0
```

Matrix B:
```
0   0   0   0
0   0   0   0
0   0   0   0
0   0   0   0
```

Strassen's Multiplication Output:
```
0   0   0   0
0   0   0   0
0   0   0   0
0   0   0   0
```
Standard Multiplication Output:
```
0   0   0   0
0   0   0   0
0   0   0   0
0   0   0   0
```

2. [30%] Implement the **tromino tiling algorithm**. Your program should take an input positive integer *k* and the position of the hole as the Linux command line and generate a $2^k * 2^k$ board. For example if your input is 4 then your code should generate 16 x16 board. ~~**Visualize the final result**, similar to the illustration in Slide 7 in Chapter 5.~~
represent you result as the below image


1	1	4	4
1	x	3	4
2	3	3	5
2	2	5	5

Here "digit" represents the regular tile cell and "X" represents a hole.

Make sure your program correctly implements the tromino tiling algorithm with $O(n^2)$ time complexity (the divide and conquer algorithm described in class); that is, there must be only one hole on the board and each of all the remaining $2^k * 2^k$ -1 squares on the board must be covered by exactly one square of one tromino tile. No credit will be given if the algorithm is incorrectly implemented, the time complexity of your program is higher than $O(n^2)$, or your program only works for specific k values.

There should be a function named **trominoTile()**, which will be called from your main() function. main() should be able to read the user input. To generate a board and print it, you can create additional function calls. Make sure you put appropriate one or two line comments to each of your function definitions. Inappropriate function declaration and unnecessary function calls will cost your points.

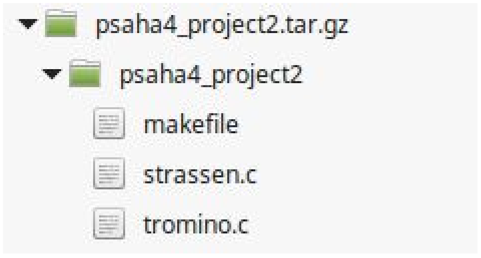
For the tromino problem, the c or cpp file name should be **tromino.c or tromino.cpp.** Your makefile should generate a .out file as tromino.out. Make sure your makefile will not run/execute the file.

To run tromino program your command should be:
./tromino.out <value of k> <hole position row number> <hole position column number>
example: ./tromino.out 4 5 6

3. [10%] Elegant, compact, and easy to read code with meaningful comments.
4. [10%] Exactly follow file naming, directory naming, packaging, and submission instructions given next.

Submission Guidelines:

Please submit a tar.gz file with the below naming convention. <user name>_project2.tar.gz
Make sure you include all the source code files as mentioned below:



```
psaha4_project2.tar.gz
    psaha4_project2
        makefile
        strassen.c
        tromino.c
```

Your makefile should be able to compile both the source code and generate .out files. make sure you makefile will not run the executable files.

You should run and test your code in **remote.cs.binghamton.edu** and make sure it works there without any issues. You code will be evaluated in the same environment. If code does not run on remote server the TA will not run it on any other environments.

Wrong directory structure and presence of unnecessary files will cost you points.

Do not create unnecessary output files that are not asked to create.

Please do not assume anything, email questions to the TA (psaha4@binghamton.edu), and see him during his office hours for any required clarifications.

Important Policies:

- **All work must represent each individual student's own effort**. If you show your code or any other part of your work to somebody else or copy or adapt somebody else's work found online or offline, you will get zero and be penalized per the Watson School **Academic Honesty Code** (http://www.binghamton.edu/watson/about/honesty-policy.pdf). To detect software plagiarism, everybody's code will be cross-checked using an automated tools. On the first violation, you will receive zero point for this project, and have to fill in and sign the official form that will be kept by the Binghamton University. On the second violation, it will be reported to the Watson School Committee for Academic Honesty for an official hearing.

- Your code will be compiled and executed. **If your code does not compile or produce any runtime errors such as segmentation faults or bus errors, you will get zero.** Before submitting your code, run it in a Linux machine (e.g., a remote machine) to remove any such errors. You can use "**valgrind**" (http://valgrind.org/) to debug memory errors, such as segmentation faults or bus errors. Note that **robust programming** is essential for developing high quality software. If you are curious about robust programming, check this out: http://nob.cs.ucdavis.edu/bishop/secprog/robust.html

- **You are supposed to figure out how to implement this project. The instructor and TA will be more than happy to explain fundamental algorithms covered in class; however, they will not teach you how to implement them or help you to make any kind of decision. Neither will they read or debug your code.** By the same token, the instructor and TA will not take a look at an emailed code. If you have questions about general C/C++ programming or Linux, use Google search, which will be much faster than asking the TA. (Don't copy from the Internet though.)