1. Goal
The goal of this assignment is to write your first real C program while admiring the cache’s impact on computation time.

2. Problem Statement [client’s statement of their need]
Mr. Krabs needs a lightning quick matrix multiplier.
Back Story: Mr. Krabs has three suppliers for crab, lettuce, sauce, and buns. Each charges different amounts for each ingredient. He also knows how much of each he uses each day. Sponge Bob told him that to compute the cost for each ingredient on each day he need only do some matrix multiplication. Mr. Krabs has hired you (and not the sponge) to code him a lightning quick matrix multiplier.

3. Analysis [What is the client’s problem?] 
Q: What is matrix? 
A: A two-dimensional array of doubles.
Q: What are the steps in multiplying two matrices? 

What to hand in
O.K., first there was a “debate.” Patrick said that he knows a better way — change the order of your outer two loops. Sponge bob said Patrick is wrong, you want to change the order of your inner two loops. Do both creating three versions of the function multiply.

(1) A well-formatted 2-up printout of your source code including all three version of multiply. You **must** use a2ps after removing all the tabs from your code. (Ask me how vi or sed can be used to accomplish this.) 

(2) A no-more-than one page write up explaining who was right and why you think it matters.

(3) **Commit and push your source code to GitHub as you work.**
   - Yes `git log` will tell me if you Hail Mary it.
   - Include a `Readme.md` file in your repository.
   - It is bad (for your grade) if you commit derivable files.
   - I expect to pull your code, run make, and then run my test script.

Assignment Requirements [part of being a course rather than software development]
- Check out the goodies at `~binkley/public_html/366/src/asn1`. Yes you should use the makefile.
- Write you code as the function
  ```c
  void multiply(array A, array B, array C, int n)
  where c = a × b and each array is a square, n×n, matrix.
  ```
- Make sure your code generates no warnings or errors when compiled using `gcc -Wall`.
• Use clear internal documentation and careful formatting. Be consistent in indentation (2 or at most 4 spaces please) and the alignment of braces. Each open brace “{” must be on its own line.

• Each source code file must start with the following header comment
  
  // This is my code
  // <Your Name>
  // CS366

• Include as a comment right after the header comment the processor model, cache size, and alignment (i.e., the block or line size), as well as the sizes of the L1, L2, and L3 caches. The first two can be obtained using the commands
  
grep model /proc/cpuinfo | head -2
  grep cache /proc/cpuinfo | head -2
  
(try these commands without the “| head -2”)

The size of the L1, L2, and L3 caches can be found on cites such as http://www.cpu-world.com

Extra Fun!

Run the command grep processor /proc/cpuinfo and then count the number of processors (or try grep processor /proc/cpuinfo | wc). Let n be this number divided by 2. Compared to running one copy of your code, is running n – 1 copies faster or slower? How about n + 1 copies?