# CS 2275 Exercise 13- Timing Code

Module 13 objectives: Timing code in cache and the CPU  
  
Readings for Module 13: Lippman 17.2, 17.4, 12, A.3

Using Dev-C++, create a .cpp source file titled ex13-<your last name>.cpp that contains the code used to solve the following problems. Use a .doc or .txt file ex13<your last name>.doc to provide the answers to any text questions. As with all assignments in this course, be sure to include a block comment just about the function indicating its purpose and use a really good function name. Include code under main to test your function by allowing the user to enter the needed parameters. Make sure you test your function sufficiently to insure correctness.

* 1. Note 13.3 contains C++ code accessing two 2D arrays using both row major and column major traversals. The C++ clock class is used to time this access. One of these arrays is implemented as an array of pointers to arrays and the other as a 1D array with a calculation used to map it as a 2D space. There is additional code for a 3rd 2D array that is commented out.
     1. Compare the times for row and column major access for the array of arrays as the sizes varies from 100x100, 1000x1000, and 10000x10000. You may need to add an additional loop to make the times long enough not to have Clock’s resolution limitations effect the results. Use the low optimization setting. Note you will need to generate three results for this test. You will need to run this test several times an taken an average.
     2. Compare the times for the 1D array mapped as a 2D array as the sizes vary using the lowest optimization setting.
     3. Uncomment the provided code to create the soo 2D “auto array” on the heap. Add code similar to that used for the other 2 types of 2D arrays to time it. Compare the access times with the other 2D arrays as the sizes vary using the lowest optimization setting.
     4. Compare i, ii, and iii using the highest optimization setting.
     5. Why is the row or col access faster? (Hint – think about cache)
     6. Which of the three types of 2D arrays is fastest and why?
     7. Why is there an extra loop going through the array prior to the timing? What happens if you do not include it?
     8. When you would use which of the three types of 2D arrays allocated on the heap?
  2. The book has a very nice discussion of pseudo random number generation (17.4) on page 745, where it recommends using the default\_random\_engine rather than rand(), the traditional C pseudo random number generator. Class note 13.5 contains the C++ code for RandFoo(), an alternate pseudo random number generator I downloaded off the internet that uses &, <<, and ^ to generate pseudo random numbers. Compare the times for rand(), the default\_random\_engine, and randFoo() using both a low and highest gcc optimization setting for generation of a large enough number of random numbers to generate significant results over the noise from the Clock class. You thus will need to generate results for each pseudo random number generator X with both the complier’s low and highest optimization settings. You will need to run each test several times and take an average to obtain a better result.
     1. Generate average results for each pseudo random number generator for both low and highest.
     2. Which is fastest?

Grading Rubric

1.i through 1.viii, 2ii -- 10 points each, 2i -- 20 points. Style up to -10: poor variable or poor function names or no block comment with a goal statement for a function. Solutions that have significantly more lines that needed will be docked points for lack of elegance.

**REMINDERS**

* code that does not compile will not be graded
* the grader should not need to modify/uncomment your code to test it. Provide a test mechanism allowing the grader to enter various tests for teach function.