CECS 625 Parallel Programming Homework Assignment #3

September 23, 2019 (100 points)

Due: October 2 (Wed) midnight

(Submit your project report and the required VS 2015 project to the Blackboard.)

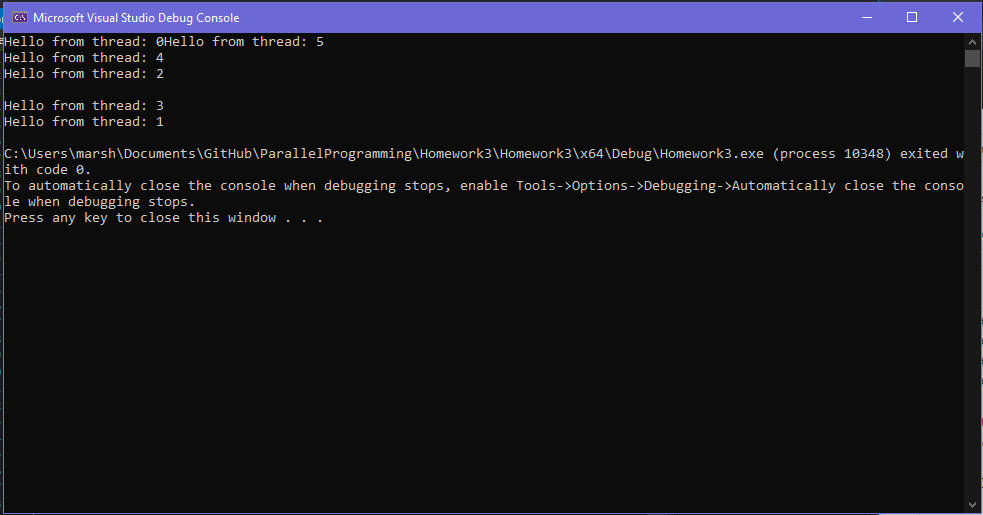
Assignment Description

For this assignment, yo need to study sections 4.1 to 4.5 (pp. 77-121) of the textbook.

Create a VS 2015 C++ console project and copy the folders data and include (which can be dowanloaded from the Blackboard to the project home folder as demonstrated in class. Use this project to do the following problems and all the required cpp files mentioned in the problems are available from the textbook’s source code website:

<https://github.com/JGU-HPC/parallelprogrammingbook/tree/master/>.

1(10 points) Section 4.1

 Run the first multithreded program, hello\_world.cpp, using 6 threads. In the project   
 repot, show the screenshot of the output and explain the output

The output shows the method say\_hello being ran on 6 simoutaneous threads. The output looks jumbled up and out of order due to all of the threads competing to access the output buffer of the compiler.

2 (20 points) Section 4.2

Test run traditional.cpp and promise\_future.cpp. In the report, explain the   
 differences of these two approachs of handling return value.

3 (35 points) Section 4.3

1. Test run matrix\_vector.cpp and compare sequential time and parallel (using cyclic\_parallel\_mult function) time. In the project report, show the timing results and explain why using lambda expressions in the implementation.
2. In the parallel implementation, cyclic\_parallel\_mult, on pages 100-101 of the textbook), replace Lines 13-22 by the following code  
     
   auto cyclic = [&] (const index\_t& id) -> void

{

for (index\_t row = id; row < n; row += num\_threads)   
 {

// initialize result vector to zero

b[row] = 0;

// directly accumulate in b[row]

for (index\_t col = 0; col < n; col++)

b[row] += A[row\*n+col]\*x[col];

}

}

Run the modified version and measure the running time and in the report, show the  
 result and explain why the time performance is worse than the original one.

4 (35 points) Section 4.4

1. Run all\_pair.cpp to collect timing results of sequential\_all\_pairs and dynamic\_all\_pairs and in the report, show the timing results.
2. In the report, explain why dynamic scheduling is better than static scheduling for the all-pair distances problem using the formula (4.6) given on page 110 of the textbook.
3. In the report, explain the key ideas of the dynamic\_all\_pairs function implementation and how and why the std::mutex object is used.