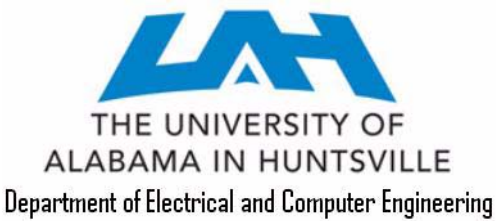

CPE 412/512 Exam 2

Fall Semester 2015
Exam 2 (Take-Home)

Name _____

INSTRUCTIONS: **CPE 512 students must work <u>all</u> five (5) problems. CPE 412 students should work any four (4) problems clearly indicating which problem they will omit. In addition to electronic submissions on the UAH Canvas course administration system student are required to submit a complete hardcopy of this exam by its due date at the beginning of class on Tuesday November 17, 2015 or submitted electronically on the CPE 512/ 412 Canvas site by Wednesday November 18, 2015 by 11:59 PM.**

Statement of Compliance with UAH Academic Misconduct Policies

I _____ certify that I have worked independently of
(sign your name)
others on this test and the work that I am presenting is my own. I am familiar with the UAH academic misconduct policy as outlined in the UAH student handbook and have agreed to abide by the policies that are stated in this document.

1. *Fully answer the following questions. Justify your answers by providing the page/section number reference from your text which supports your answer or by providing a complete citation of any external sources that you have used.*

a) What OpenMP directive is used to provide mutual exclusion synchronization in sections of code? Give an example of its use and explain why it is needed.

b) Give the output of the following OpenMP code fragment if possible or discuss why this is not allowed. What is the effective scope of the sum variable? How could the answer be affected if the reduction clause is not used? Assume that there are 4 threads all together:

```
int i; double sum = 0.0;
#pragma omp parallel for reduction(+:sum)
for (i=1; i <= 4; i++)
      sum = sum + i;
cout << "The sum is " << sum << endl;
```

c) Research how you can create MPMD code using MPI on the dmc.asc.edu system. List the steps you would take to use two separate source files that execute as two different MPI processes but can communicate with one another using standard MPI functions such as MPI_Send and MPI_Recv.

*d) For the following **pthread** code what is (are) the possible value(s) of num if there are no assumptions regarding the targeted system or the manner in which the **pthread** scheduler will operate? Justify your answer.*

```
using namespace std;
#include <iostream>
#include <pthread.h>
#include <stdlib.h>

int num=1;
void *thread1(void *dummy) {
   num = num + 1;
}
void *thread2(void *dummy) {
   num = num * 3;
}
void *thread3(void *dummy) {
   num = 123;
}

int main(int argc, char *argv[]) {
   pthread_t thread1_id,thread2_id,thread3_id;
   pthread_create(&thread1_id,NULL,thread1,NULL);
   pthread_create(&thread2_id,NULL,thread2,NULL);
   pthread_create(&thread3_id,NULL,thread3,NULL);
   cout << "num = " << num << endl;
   while(1); // infinite loop
}
```

e) What is the difference between local and global synchronization. Give an example of a global synchronization construct. Why is local synchronization preferred to global synchronization methods when both techniques can be applied in a manner that does not affect the correctness of the code?

f) Explain why static mapping of data blocks to processors may be bad for the Mandelbrot program that is discussed in Lecture 20.

g) What is the basic idea of a Monte Carlo simulation discussed in Lecture 20. Describe how this technique can be used to solve the numerical integration problem? What makes this technique so attractive for parallel processing? What are the major issues that could affect its accuracy when it is parallelized?

2. *You have been provided with a sequential traveling program that you are to parallelize using OpenMP, or pThreads. This sequential program which is named* ***tsp_serial.cpp****, can be found on the CPE 512/412 Canvas™ site.*

   a) Examine this program and modify it so that you can measure the execution time. Using the queuing system on the dmc.asc.edu system, as discussed in class, record the execution times for a 3,4,5,6,7,8,9,10,11,12,13, and 14 city tour. What is the time complexity of this algorithm? Develop an equation that can be used to estimate the execution time of the base sequential algorithm. Why do you think this problem is a challenging problem to solve, and inexact heuristics are utilized instead of exhaustively searching through all solutions as is done in this program?

   b) Develop a general multi-threaded version of the sequential program in either pThreads or OpenMP that will effectively divide up the amount of work that is performed. Using the dmc.asc.edu queuing system, measure the execution times for an 8,9,10,11,12, 13 and 14 city tour on a 2, 4, 8, and 12 thread implementation. For each multi-threaded implementation show the speedup, efficiency, and cost as a function of the number of cities in the tour.

3. *Expand the* ***mm_mult_serial.cpp*** *program on the CPE 512/412 Canvas TM site (or copy this program from /home/shared/wells/exam2/mm_mult_serial.cpp on the dmc.asc.edu system) to create a hybrid multi-threaded/message passing implementation of a matrix/ matrix multiplication program where the first matrix is of size* ***lxm*** *and the second matrix is of size* ***mxn****. Assume that the quantities being multiplied are of type* ***float****. Write the program in a general manner to allows the number of message passing processes and the number of threads per message passing process to be independently set by the user at run time. The program should be designed using the same row-wise decomposition method that was used in the two homework assignments. It should be written in a manner that will result in the total amount of computation to be divided as evenly as possible among the message-passing processes with the computation that is assigned to each message-passing processes in turn being further divided as evenly as possible between the associated threads. You are to use a combination of* ***MPI*** *and either* ***OpenMP*** *or* ***pThreads*** *to complete this problem. The number of threads should be a command-line parameter while the number of message-passing processes should be set using the* ***mpirun*** *command. The l, m, and n dimensions should also be command line parameters that can be set at run time. If the executable is named* ***mm_mult_hybrid*** *then the syntax needed to execute the code should take on the general form shown below:*

   mpirun -np [No. Processes] mm_mult_hybrid [No. Threads/Process] [dim_l] [dim_m] [dim_n])

   a) Illustrate the correctness of your program for all combinations of thread numbers and number of processes whose product is equal to the value of 8 for cases where ***dim_l****,* ***dim_m****, and* ***dim_n*** *take on distinct values.*

   b) Then executing the program on the main interactive node of the the dmc.asc.edu system, as discussed in class, record the execution times under the same conditions (process thread product equal to 8) but with square matrices of size ***m=n=l=****5,000****. Do this at least 5 times for each thread/process combination reporting the minimum execution time that you obtained in your repeated runs for each case. After this compare these execution times with that of the original serial program. What is the relative speedup and efficiency for each case. Is there a significant difference between the various parallel implementations? If so give a possible explaination as to why the execution time was not the same for each case given that the total number of threads is the same in all cases and your code was designed to evenly distribute the workload among the threads.*

4. *Write a parallel program to compute the definite integral*

$$\int_{0.01}^{1} \left(x + \sin\frac{1}{x} \right) dx$$

using the trapezoidal decomposition method discussed in Lecture 21. Write this program in a generic manner using either pthreads, or OpenMP (just one). The program should allow the user to set the number of threads at runtime (i.e. one or more) as a command line parameter. The total number of computational intervals should also be able to be set by the user a run-time using a command line parameter. (The range over which integration occurs can be hard-coded or can be set using two additional command line parameters.) The program should be designed in a manner where it divides the number of computational intervals per thread as evenly as possible to achieve the best processing load balance between the threads.

5. *Expand upon the* ***bounded_buffer.cpp*** *that is provided on Canvas (or copy this program from /home/shared/wells/exam2/bounded_buffer.cpp on the dmc.asc.edu system) to create two separate OpenMP implementations of the producer/consumer bounded buffer-problem discussed in Lecture 21 that utilizes a common shared memory (between threads) and counting semaphores to ensure proper synchronization. [Note you will also need to link this program with the* ***util.o*** *file that is also provided on Canvas or on the dmc.asc.edu system at /home/shared/wells/exam2/util.o] One implementation should us the standard parallel and work sharing data parallel constructs to start up the separate consumer and producer threads. The other implementation should utilize the OpenMP tasking model. Verify that both models function correctly and answer the following questions:*

   a) Carefully explain the characteristics of the producer and consumer bounded buffer problem? What are the major synchronization challenges that are involved.

   b) Describe the function of the main, consumer, and producer module? What are the output files diary and con_? and prod_? reporting?

   c) Execute both versions of the producer/consumer program with two producers, two consumers, number of messages of 100 and with a buffer size of 5. How many actual threads are generated when you run each version? Are they the same? How do they relate to the number of producers/consumers that are specified at run time?

   d) How is it possible for the buffer size to be smaller than the number of data that is to be passed between producer and consumer? Explain how this is actually implemented in the program.

   e) Experiment with different size buffers, messages, and consumer processes and producer processes. From the diary file and other files can you determine if the semaphore releasing strategy is fair? Explain your answer.