ECE 563 Midterm 2
Spring 2015

This exam is open book and open notes. Please print and sign your name below. By doing so you signify that you have not received or given any prohibited help on this exam.

Answer all questions on the answer sheet that is the last sheet of this test. You may detach the answer sheet when taking the test and you will turn it in separately. There are 23 questions in this test and all are worth an equal number of points.

Name:

The code below is used for Questions 1, 2 and 3.

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]) {

  int i;

  omp_set_num_threads(4);

  #pragma omp parallel private(j)
  {
     #pragma omp for
     for (i=0; i < 10; i++) {
       printf("a\n");
     }

     #pragma omp critical
     for (i=0; i < 10; j++) {
       printf("b\n");
     }

     #pragma omp single
     for (i=0; i < 10; i++) {
       printf("c\n");
     }
  }
}
```

Question 1. How may a's are printed by the program?

(a) 1 a's
(b) 4 a's
(c) 10 a's
(d) 40 a's

Question 2. How may b's are printed by the program?

(a) 1 b's
(b) 4 b's
(c) 10 b's
(d) 40 b's

Question 3. How may b's are printed by the program?

(a) 1 c's
(b) 4 c's
(c) 10 c's
(d) 40 c's

Question 4 - 9.
Consider an MPI program that uses a Reduce of **n** numbers followed by a Bcast of the result of the Reduce to all P processes. During the reduction phase, count any arithmetic as well as communication that occurs as taking 1 unit of time.

For each question below, pick the best answer.

Question 4. The sequential time is:
(a) n
(b) 2n

Question 5. The parallel work is:
(a) $n/P + 3 \log_2 P$
(b) $n/P + 4 \log_2 P$
(c) $2n/P + 3 \log_2 P$
(d) $2n/P + 4 \log_2 P$

Question 6. The parallel overhead is:
(a) $P(n/P + 3 \log_2 P) - n$
(b) $P(n/P + 4 \log_2 P) - n$
(c) $P(2n/P + 3 \log_2 P) - 2n$
(d) $P(2n/P + 4 \log_2 P) - 2n$

Question 7. The isoefficiency function is:
(a) $3P \log_2 P$
(b) $4P \log_2 P$

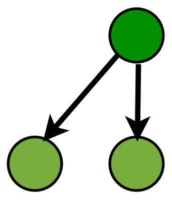
Question 8. The units of work needed on 4 and 16 processors to maintain the same efficiency is:
(a) 24 and 192
(b) 32 and 256

Question 9. If the efficient MPI_AllReduce discussed in class is used, the parallel work will be:
(a) $n/P + 2 \log_2 P$
(b) $n/P + 3 \log_2 P$
(c) $2n/P + 2 \log_2 P$
(d) $2 n/P + 3 \log_2 P$

Consider the Pthreads program below:

```
void foo(void* p) {
   int left, right;
   left = right = 0;
   if (p) {
      pthread_t leftT, rightT;
      int lrc = pthread_create(&leftT, NULL, foo, (void *) ((Node*) p)->left);
      int rrc = pthread_create(&rightT, NULL, foo, (void *) ((Node*) p)->right);
      pthread_join(leftT, (void *) &left); // A
      pthread_join(rightT, (void *) &right); // B
   }
   pthread_exit((void*) 1 + left + right);
}

int main(....) {
   Node* r;
   int v;
   // allocate binary tree with root r;
   int rc = pthread_create(&rootT, NULL, foo, (void *) r);
   pthread_join(rootT, (void *) &v);
   printf("%d found: %d\n", v);
   pthread_exit(0);  // C
}
```



Question 10. How many threads are created during all calls to **foo (a)**?
(a) 1
(b) 2
(c) 3
(d) 4
(e) 5
(f) 6
(g) 7

Question 11. Pick the answer that is most true:
(a) Statement B will complete as soon as the rightT thread finishes, even if the leftT thread is still executing
(b) Statement B will complete after threads rightT and leftT finish
(c) Statement B will check if thread rightT has finished, and if it has not the argument **right** will contain an error return code.

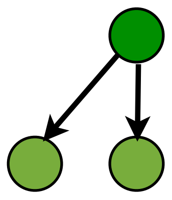
Question 12: Pick the most correct answer about the pthread_exit statement at the end of the main (see comment C)
(a) the pthread_exit at the end of a main routine allows all threads created in the program to continue executing after main exits
(b) the pthread_exit in this main will not execute until all threads created in foo finish
(c) pthread_exit allows a thread to return a value.
(d) all of the above.

Question 13: True or false: only attached threads can be joined.

Consider the OpenMP program below:

```
int foo(Node* p) {
   int left, right;
   left = right = 0;
   if (p) {
         #pragma omp task
         left = foo(p->left);
         #pragma omp task
         right = foo(p->right);
          #pragma omp taskwait // A
   }
   return 1 + left + right; // B
}

int main(....) {
   Node* r;
   int v;
   // allocate an initialize a binary tree with root r;
   #pragma omp parallel
   {
      #pragma omp single
      {
         v = foo(r);
      }
   }
   printf("%d found: %d\n", v);
}
```



Question 14. How many tasks are created during all calls to **foo (a)**?
(a) 1
(b) 2
(c) 3
(d) 4
(e) 5
(f) 6
(g) 7

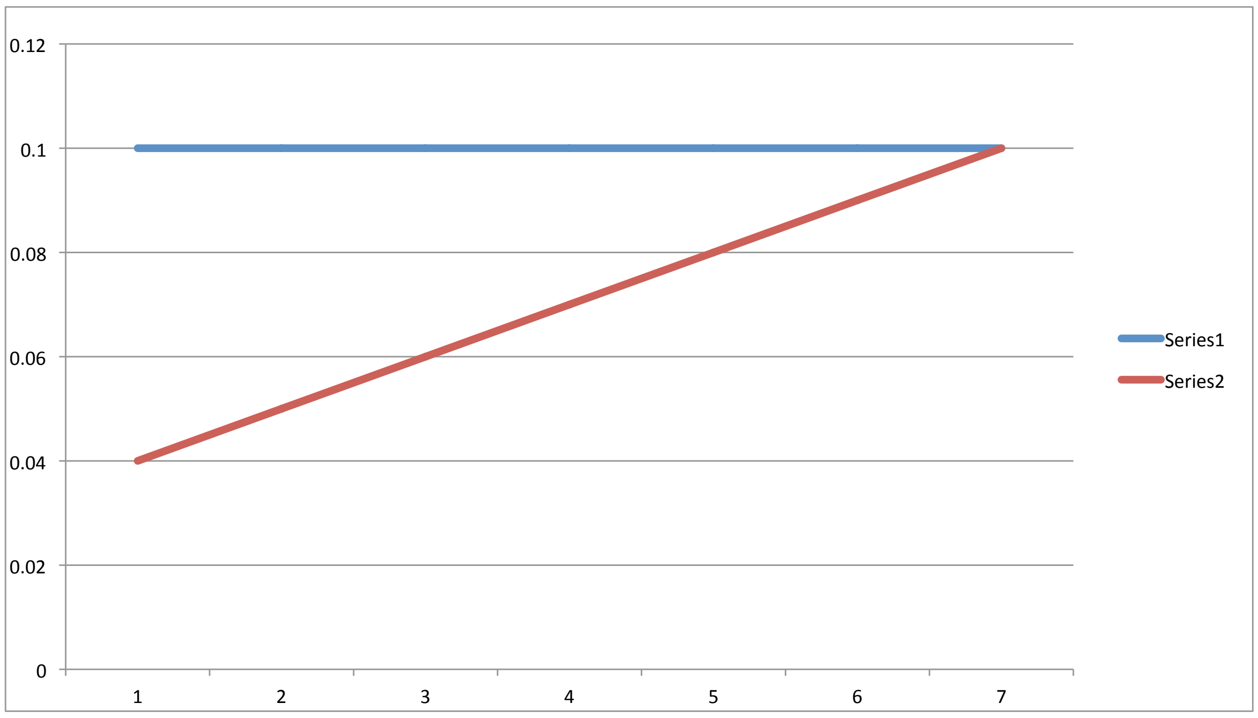
Question 15. Answer true or false. If the bold **#pragma omp taskwait** statement (statement A) were removed, then left and right would still have the return values from the foo calls when the bold **return** statement (statement B) was executed.

Question 16. Answer true or false. The number of tasks cannot exceed the number of threads.

Question 17. Ragu's final 563 project achieves a speedup of 8 on twelve processors. Using Amdahl's law find *f*, the serial fraction of the program

Question 18. What is the efficiency of a program that has a speedup of 8 on 12 processors?

Question 19. Given the experimentally determined serial overheads e as shown below, which program is likely to give better performance as the number of processors is increased?
(a) The program whose e is shown by series 1
(b) the program whose e is shown by series 2



Question 20. An application executing on 100 processors requires 75 seconds to run. It is experimentally determined through benchmarking that 10% of the time is spent in the serial code on a single processor. What is the scaled speedup of the application?

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]) {

  int i;
  int a[1000], b[1000], c[1000], d[1000], idx[10];
  omp_lock_t locks[10];

  #pragma omp parallel for
  for (i = 0; i < 1000; i++) {
      // init element i of a, b, c and d to i
      a[i] = b[i] = c[i] = d[i] = i;
      idx[i] = i % 10;
  }

  for (i = 0; i < 10; i++) {
      omp_init_lock(&locks[i]);
  }

  #pragma omp parallel
  {
     #pragma omp for nowait
     for (i = 0; i < 1000; i++) {
        a[i] = b[i] + c[i];
     }

     #pragma omp for
     for (i = 0; i < 1000; i++) {
        int t = idx[i];
        omp_set_lock(&locks[t]);
        c[t] = a[i-1] + d[i];
        d[i-1] = a[i];
        omp_unset_lock(&locks[t]);
     }
  }
}
```

Question 21. True or false: there is a race on one or more elements of the **a** array.

Question 22. True or false: there is a race on one or more elements of the **c** array.

Question 23. True or false: there is a race on one or more elements of the **d** array.

This page intentionally left blank

Answer sheet

This exam is open book and open notes and no electronics. Please print and sign your name below. By doing so you signify that you have not received or given any prohibited help on this exam.

Name:

1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.