UNIVERSITY OF WATERLOO

Final Examination

Term: Winter **Year:** 2011

Student Name: _____

Student Identification Number: _____

Course Abbreviation and Number:	ECE459
Course Title:	Programming for Performance
Section(s):	001
Sections Combined Course(s):	
Section Numbers of Combined Course(s):	
Instructor(s):	Patrick Lam
Date of Exam:	April 18, 2011
Exam Period Start time: 9:00	End time: 11:30
Duration of Exam:	2.5 hours
Number of Exam Pages: (includes cover page)	5

Exam Type: (select one) ☐ Closed Book ☐ Special Materials ☒ Open Book

Materials Allowed: (select one)

☐ No additional materials are allowed.

☒ Materials allowed are listed below:

Any printed materials. No electronic devices.

Exams are printed double sided on white paper.

☐ Select this box if second side of paper is to be used for rough work calculations.

Marking Scheme:

Question	Score	Question	Score
1		6	
2		7	
3		8	
4		9	
5		10	

Programming for Performance (ECE459): Final

April 18, 2010

This open-book final has 6 questions, worth 20 points each. Answer all questions. Write the answers in your answer book. You may consult any printed material (books, notes, etc).

Question 1: Short Answer

One point each. **Answer these questions in your exam booklet.**

(a) Two library functions for doing communication in MPI are _____ and _____.

(b) Doing more work in parallel increases a system's _____.

(c) Modern CPUs spend most of their time waiting due to _____.

(d) When using OpenCL, you may choose to divide the computation space into _____.

(e) Gustafson's Law says that parallelization isn't hopeless when you can increase the _____.

(f) One condition that would impede automatic parallelization is _____.

(g) One primary design goal of DTrace was _____.

(h) A different primary design goal of DTrace was _____.

(i) An `sfence` instruction prevents reordering of _____.

(j) To effectively parallelize an OpenMP loop where different iterations run for different amounts of time, you want to use _____.

(k) `oprofile` is an example of a _____ profiler.

(l) The three steps in using profile-guided optimization are: _____.

(m) The main difference between combine and reduce in MapReduce is that _____.

(n) The obvious way to get rid of a race condition is by using _____.

(o) A technique to get rid of a WAR dependency is _____.

(p) The first thing you should do before trying to improve performance of some code is to _____ it.

(q) The term for speeding things up by doing many things at once is _____.

(r) Re-entrancy for a lock means that you can _____.

(s) Botnets are a good example of this parallelization pattern: _____.

(t) The bit of code that runs massively parallel in GPU programming is a _____.

Question 2: OpenMP

Consider the following code[1].

```
1  # define NV 4
2
3  /* don't worry about mind, connected */
4  # pragma omp parallel /* private, shared etc */
5  {
6    my_id = omp_get_thread_num ( );
7    nth = omp_get_num_threads ( );
8    my_first =   (    my_id       * NV ) / nth;
9    my_last  =   ( ( my_id + 1 ) * NV ) / nth - 1;
10   # pragma omp single
11   {
12     printf (" P%d: Parallel region begins with %d threads\n",
13             my_id, nth);
14     printf ("\n");
15   }
16   fprintf (stdout, " P%d:  First=%d  Last=%d\n",
17            my_id, my_first, my_last);
18
19   for ( my_step = 1; my_step < NV; my_step++ )
20   {
21     # pragma omp single
22     {
23       md = i4_huge;
24       mv = -1;
25     }
26     find_nearest (my_first, my_last, mind, connected, &my_md, &my_mv);
27     # pragma omp barrier
28   }
29 }
```

(a) Explain what each of the OpenMP pragmas does.

(b) Assume that `OMP_NUM_THREADS` is 4 and draw a diagram explaining what the threads do.

[1] `http://people.sc.fsu.edu/~jburkardt/c_src/dijkstra_open_mp/dijkstra_open_mp.html`.

Question 3: Reductions

We saw this example of a reduction in the notes.

```
1  double sum (double *array, int length)
2  {
3    double total = 0;
4
5    for (int i = 0; i < length; i++)
6      total += array[i];
7    return total;
8  }
```

I mentioned that the Solaris compiler could detect that it was a reduction and parallelize it.

(a) Write down, in reasonably-detailed pseudocode, the corresponding parallelized code.

(b) Write down the assumptions that you're making about the `+` operator in your parallelization.

Question 4: Memory Barriers and Consistency Models

Consider the following code; all variables are initially 0.

```
T1: x = 1; r1 = y;
T2: y = x; r2 = x;
```

Assume the architecture is not sequentially consistent.

- Show me all possible (intermediate and final) memory values and how they arise.

Question 5:

Here is some C code from meschach.

```
1  double   zm_norm1(ZMAT *A)
2  {
3       int i, j, m, n;
4       Real maxval, sum;
5
6       if (A == ZMNULL)
7            error(E_NULL,"zm_norm1");
8
9       m = A->m; n = A->n;
10      maxval = 0.0;
```

```
11
12      for (j = 0; j < n; j++)
13      {
14          sum = 0.0;
15          for ( i = 0; i < m; i ++ )
16              sum += zabs(A->me[i][j]);
17          maxval = max(maxval,sum);
18      }
19
20      return maxval;
21  }
```

- Describe 2 compiler optimizations that could apply to the code and the resulting code after optimization. Briefly summarize the conditions that need to hold for the optimizations to be safe; you may add qualifiers to the code if you want.

Question 6: GPU Programming

Here is some C code[2].

```
# define M 500
# define N 500

  int i, j;
  double diff;
  double u[M][N], w[M][N];

  diff = 0.0;
  for ( i = 1; i < M - 1; i++ ) {
    for ( j = 1; j < N - 1; j++ ) {
      w[i][j] = ( u[i-1][j] + u[i+1][j] + u[i][j-1] + u[i][j+1] ) / 4.0;

      if ( diff < fabs ( w[i][j] - u[i][j] ) )
        diff = fabs ( w[i][j] - u[i][j] );
    }
  }
```

- Express the code as one or many OpenCL kernels. (I don't care about host code, just the kernel itself.)
- Indicate the floating-point operations in the kernel.
- Do you need to worry about synchronization?

[2] http://people.sc.fsu.edu/~jburkardt/c_src/heated_plate/heated_plate.c