ECE 563 Second Exam, Spring 2014

Don't start working on this until I say so

Your exam should have 8 pages total (including this cover sheet) and 11 questions. Each questions is worth 9 points. Please let Prof. Midkiff know immediately if it does not. **Show your work.**

This exam is open book and open notes. **No electronics are allowed.** If you have a question, please ask for clarification. If the question is still not clear, state whatever assumptions you need to make to answer the question, and answer it under those assumptions. Check the front board occasionally for corrections.

Name (print clearly):

n	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
$\log_2 n$	1	2	3	4	5	6	7	8	9	10

Question 1.

The sequential version of a program executes in 10000 hours. Executing on 1000 processors it executes in 20 hours.

What is the speedup and efficiency?

How long would the program take to execute if the efficiency when executing on 1000 processors was 100%?

Question 2.

Gene needs to run a simulation for his thesis and his advisor wants him to run it for a fixed problem size. Gene can make 90% of the program parallel, with 10% of it being sequential.

What speedup can Gene expect on 10 processors?

What would be the maximum speedup on an infinite number of processors?

Question 3.

Mary has a problem whose size can increase with an increasing number of processors. She executes the program and determines that in a parallel execution on 100 processors. 5% of the time is spent in the sequential part of the program. What is the scaled speedup of the program on 100 processors?

Question 4.

Gene has another program that scales with increasing problem size, but as he executes it on larger numbers of processors he notices that the speedup is not what he expected. He performs a Karp-Flatt analysis and notices that the *experimentally determined serial fraction e* is increasing with an increasing number of processors. This means that (circle the one that is most correct)

The lack of speedup is entirely caused by inherently parallel portions of the algorithm

The lack of speedup is caused, at least in part, by overheads that increase with the machine size

Karp-Flatt analysis is not useful for diagnosing the cause of performance issues when executing on increasing numbers of processors.

Question 5.

Josephine writes a program where the problem size scales with the an increasing number of processors. The cluster she runs on has 4GB per node memory and executing on 8 processors she is using 1GB of memory. She performs an isoefficiency analysis on the program and determines that the isoefficiency relationship is $W = P \log_2 P$. Can she scale to 64 nodes with constant efficiency?

Question 6.

Part of the reason Josephine's program has an isoefficiency of $W = P \log_2 P$ is that she has a section of code that looks like:

```
for (i=0; i < numProcs; i++) { // numProcs is the number of processors
  MPI_Reduce(b, c, n, MPI_INT, MPI_SUM, i, MPI_COMM_WORLD);
}
```

Rewrite this to be a more efficient operation (the loop can be replaced by a single MPI collective communication call)

Question 7.

Gene is trying to improve the performance of his program and comes across the following loop:

```
// find the sum of integers in the part of the c array that is on this processor
int localSum = 0;
for (i=0; i < myUB; i++) {
  localSum += c[i];
}
// The first table below is the value of variables here (program Point A)

int recvBuf;
MPI_Status stat;
if (myRank == 0) {
   (void) MPI_Send(&localSum, 1, MPI_Int, 1, MPI_ANY, MPI_COMM_WORLD);
} else {
   (void) MPI_Recv(&recvBuf, 1, MPI_Int, myRank-1, MPI_ANY, MPI_COMM_WORLD,
                           &MPI_Status);
   localSum += recvBuf;
    if (myRank < numProc-1) {
       (void) MPI_Send(&localSum, 1, MPI_Int, myRank+1, MPI_ANY,
                                  MPI_COMM_WORLD);
    }
}
// The second table below is the value of variables here (program Point B)
```

A. Fill in the boxes below for each processor after the `for i` loop (program Point A)

	P = 0	P = 1	P = 2
c	0, 1, 2, 3	4, 5, 6, 7	8, 9, 10, 11
localSum			

B. Fill in the boxes below for each processor after the `if` (program Point B)

	P = 0	P = 1	P = 2
c	0, 1, 2, 3	4, 5, 6, 7	8, 9, 10, 11
localSum			
recvBuf	Leave Empty		

C. What MPI collective operation performs this communication pattern?

CI.Question 8.

A programmer who is worried about races uses a `critical` section to synchronize the loop below:

```
#pragma omp parallel for
for (i = 0; i < n; i++) {
#pragma omp critical
  sum = sum + data[i];
}
```

The program does not get any speedup.

Circle all that are true:

The program still has a race and the race slows down the program.

The program does not get any speedup because the critical pragma only allows one iteration to execute the body of the loop at a time.

The program could have used an OpenMP reduction to achieve the same result and have been faster.

Question 9.

Given the loop:

```
#pragma omp parallel for
for (i=1; i < n; i++) {
  a[i] = a[i-1] + i;
}
```

What references are involved in races?

i in or (i=1; i < n; i++)

i in +i and in subscripts

a[i-1]

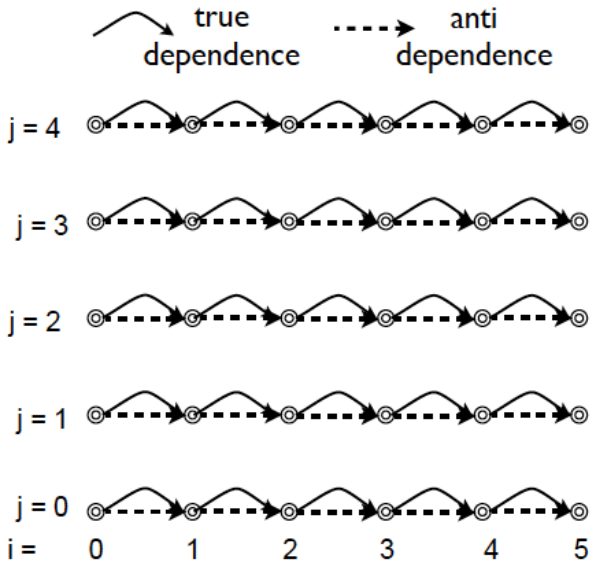
a[i]

Nothing, as no data is shared between iterations of the parallel loop

Question 10.

Given the loop nest and its dependence graph

```
for (i=1; i < n; i++) {
  for (j = 0; j < n; j++) {
    a[j][i] = (a[j][i-1] + a[j][i] + a[j][i+1])/3);
  }
}
```



Can the i loop be parallelized?

Can the j loop be parallelized?

Question 11.

Given the loop nest of Question 10,

The access order for the first 20 accesses in the a array for the loop as written above is:

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

i.e., the element with an 11 in it is the 11th element accessed.

If the i and j loop are interchanged, giving the loop:

```
for (j=1; j < n; j++) {
  for (i = 0; i < n; i++) {
    a[j][i] = (a[j][i-1] + a[j][i] + a[j][i+1])/3);
  }
}
```

The access order for the first 20 accesses is:

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

Circle the access order with the *best* cache locality.