Last 4-digits of Student ID: _______________

Northwestern University
Department of Electrical and Computer Engineering
ECE 358 Introduction to Parallel Computing

Question	Score
1	/ 20
2	/ 20
3	/ 30
4	/ 35
Total	/ 105

NOTE: Please show your work clearly for all the questions. Use the space provided for your answers as much as possible. If necessary feel free to use the back of the sheets and/or additional sheets.

Question 1) [20 pts.]

a) [5 pts.] Consider the following code segment:

```
int a[100];

int main()
{
...
}
```

What is the total number of integers that will be allocated (sum of integers allocated on all processors) if this code is executed

a) on a shared-memory machine with 4 processors,
b) on a distributed-memory machine with 4 processors.

a) 100
b) 400

b) [5 pts.] Consider the following loop:

```
for (i = 4; i < N; i++)
{
      a[i] = a[i-4] + b[i] + 2;
}
```

What is the best scheduling technique assuming that there are 4 processors in the system?

Static interleaved scheduling

c) [5 pts.] How can you perform a one-to-all broadcast using a one-to-all scatter call?

You generate an array of size p. Then, you copy the element p times in this array and perform the scatter call.

d) [5 pts.] Consider the following loop:

```
for (i = 1; i < n; i++)
{
      a[i] = b[i] + 5;
      b[i] = a[i-1];
}
```

Which one of the following optimizations is likely to result in the best performance: loop distribution, loop reordering, indirect indexing, closed form substitution?

Loop distribution

Question 2) (20 points)

```
for (i = 1; i < n; i+=2)
   a[i] = b[i] + 3;
```

Apply the following scheduling approaches to the above loop.

a) [4 points] Prescheduling
b) [4 points] Static blocked scheduling
c) [4 points] Static interleaved scheduling
d) [4 points] Dynamic self scheduling
e) [4 points] Dynamic grouped sweeping scheduling (GSS)

You can make any assumption about the number of processors and the loop size n for the execution.

```
a)
if (myid ==0)
{
for (i = 1; i < 20; i+=2)
   a[i] = b[i] + 3;
}
else if (myid == 1)
   {
   for (i = 20; i < n; i+=2)
      a[i] = b[i] + 3;
   }

b)
lower_bound = myid * (n / num_procs);
upper_bound = (myid + 1) * (n / num_procs);

for (i = lower_bound; i < upper_bound; i++)
   a[i] = b[i] + 3;


c)
for (i = myid; i < n; i+=num_procs)
   a[i] = b[i] + 3;


d)
chunk = 2;
local_i = 0;
while (local_i < n)
{
      m_lock();
            local_i = global_i;
            global_i += chunk;
      m_unlock();
      for (i = local_i; (i < local_i + chunk) && (i < n); i++)
         a[i] = b[i] + 3;
}
```

```
e)
local_i = 0;
while (local_i < n)
{
      m_lock();
            local_i = global_i;
            chunk = (n - global_i) / 8;
            global_i += chunk;
      m_unlock();
      for (i = local_i; (i < local_i + chunk) && (i < n); i++)
         a[i] = b[i] + 3;
}
```

Question 3) (30 points)

Find an approximation for the speed-up, efficiency, and Amdahl's fraction α in terms of **input size n** and the **number of processors p** for the following code segment.

```
int x[n];
int y[n];

void main()
...


for (j = 0; j < n; j++)
{
   for (i = myid; i < n; i += p)
   {
      local_num += j * y[i];
   }
   m_lock();
      x[j] += local_num;
   m_unlock();
}

...
```

Assume a constant delay of 1 for all arithmetic and memory operations. Note that the corresponding sequential code is:

```
int x[n];
int y[n];

for (j = 0; j < n; j++)
{
   for (i = 0; i < n; i++)
   {
      x[j] += j * y[i];
   }
}
```

Assuming that there are two operations in the loop, the serial time is: $2 * n^2$

With the same assumption, the parallel execution time is $[2 * n^2 / p + n * p]$, because the outer loop will be executed n times (hence n * p), and the inner loop will be executed n / p times (hence $2 * n^2 / p$).

Hence speed-up is:

$S = 2 * n^2 / [2 * n^2 / p + n * p]$

Similarly, the efficiency is:

$E = 2 * n^2 / (p * [2 * n^2 / p + n * p]) = 2 * n^2 / [2 * n^2 + n * p^2]$

Amdahl's fraction α can be found by:

Sp $= 2 * n^2 / [2 * n^2 / p + n * p]$ **| multiply by 'p / 2*n²'**

$= p / [1 + (np^2 / 2n^2)]$

if we set this to 'p / [1 + (p-1) * α(n,p)], we find

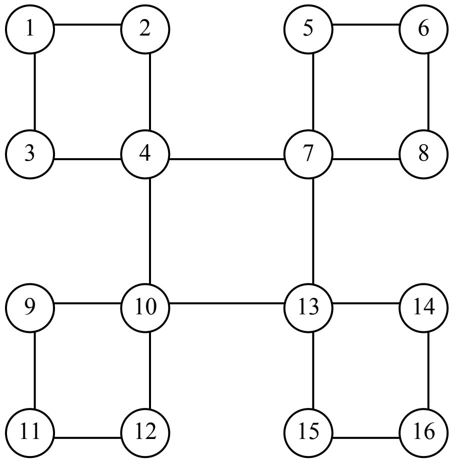
$\alpha (n,p) = (np^2 / 2n^2) / (p-1)$

$= p^2 / 2*(p-1)*n$

Question 4) (35 points)

There are a number of basic topology types such as hypercube, mesh, torus, bus, etc. Due to implementation costs, it is common to implement a combination of these in a single system. In this problem, you will analyze such a system. The system in question is built by combining rings. Particularly, the system includes M rings. Each ring has N nodes in it. In each sub-ring, one dedicated processor is used for communicating to any external processor. These dedicated processors are connected through another ring.

An example system is shown in the following figure. In this system, there are 4 rings each with 4 nodes (hence a total of 16 processors). In each subsystem (i.e., ring), there is a dedicated processor to perform any communication to the external nodes. These processors are connected through another ring.



a) [5 pts.] What is the diameter, arc connectivity, and bisection width of a system with m rings with n processors in each? Note that similar to the above figure, in each of the m rings, one of the n processors is dedicated to external communication and these dedicated processors are connected in a ring topology.

The diameter is from processor 1 to 16, which is roughly $\frac{n}{2}+\frac{m}{2}+\frac{n}{2}$ **= n + (m / 2)**

Arc connectivity and bisection width are 2.

b) [10 pts.] What is the best way to perform an all-to-all scatter operation assuming the communication is store-and-forward?

The best way is first perform an all-to-one gather (single node gather) in the small rings to gather all the "external" data elements in the communication processors. Then, perform an all-to-all scatter in the inner ring to distribute all the elements and finally perform a one-to-all scatter in the outer rings to distribute the data.

c) [5 points] What is the number of steps in the above all-to-all scatter assuming that there are m small rings with n nodes each in the system (a total of m*n processors)?

The first step will take (n / 2) iterations. The second step requires (m – 1) iterations and the final step need (n / 2) steps. So, the total number of steps is

$$\mathbf{n / 2 + m - 1 + n / 2 = n + m - 1}$$

d) [15 points] Find an estimate for communication time in terms of t_s and t_w.

Since we calculated the number of steps in the previous section, the t_s can be easily found:

$$t_s * [n + m - 1]$$

To find the cost of t_w, we need to find the amount of data sent in each step. Assuming that each element is 1 unit:

For the first step, a processor sends (m-1)n elements (in other words all the data that will be shipped to other rings; the communication across the nodes within a ring can be overlapped with the next step)

So, the total cost of this step is $(m-1)*n*(n / 2) = (m-1) n^2 / 2$

For the second step, each processor initially starts with "$(m-1)n^2$" elements (because from each processor in the outer ring, they have (m – 1)*n elements. First they send all the elements, then they send $(m-2)*n^2$ elements, then $(m-3)*n^2$ elements, ... n^2 elements. Hence, the sum of all elements is:

$n^2 * [(m - 1) + (m - 2) + ... + 1] = n^2 * m * (m - 1) / 2$

The final step is a one-to-all scatter. First, the communication processor sends (m – 1)*n / 2 to one side. Then, it sends (m – 1)* ([n / 2] – 1) to the other side (while the first node is sending (m – 1)* ([n / 2] – 1) to its neighbor, etc. So the total communication is

$(m - 1) * (n / 2) + (m - 1) * [(n / 2) - 1] + (m - 1) * [(n / 2) - 2] + ...$

$= (m - 1) * [(n/2) + ((n/2) - 1) + ((n/2) - 1) + ... + 1]$

$= (m - 1) * [n/2 * ((n / 2) + 1) / 2]$

Hence, the total cost is:

$$T = t_s * [n + m - 1] + t_w * [(m-1) n^2 / 2 + n^2 * m * (m - 1) / 2 + (m - 1) * [n/2 * ((n / 2) + 1) / 2]]$$