Problems 1, 3, and 4 will be graded. There are 65 points on these problems. Note: You must do all the problems, even the non-graded ones. If you do not do some of them, half as many points as they are worth will be subtracted from your score on the graded problems.

**Problem 1.** (20 points) Assume a two-processor system executing the following code on CPUs $P_1$ and $P_2$ respectively.

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>$i = 1$</td>
<td>2a</td>
</tr>
<tr>
<td>1b</td>
<td>$j = 1$</td>
<td>2b</td>
</tr>
<tr>
<td>1c</td>
<td>$X = k$</td>
<td>2c</td>
</tr>
</tbody>
</table>

Suppose that the initial value of every variable is 0.

(a) Assuming that the memory is sequentially consistent (see Lecture 13), list all possible $(X, Y, Z)$ triples at the end of execution of the above code by the respective processors on the two-processor machine.

(b) Total store ordering (TSO) allows reads to bypass writes. What output triples $(X, Y, Z)$ are possible with TSO but not with sequential consistency? Explain briefly why these triples are possible with TSO but not with Sequential Consistency for the above code.

(c) What is the minimum number of `membar` instructions (memory barrier, or fence; see VBEE Lecture 21, on-campus Lecture 24) that need to be inserted into the TSO model for the above code in order to guarantee sequential consistency? Show the resulting code.

(d) Partial store ordering (PSO, see Culler, Singh, and Gupta; p.689) not only allows reads to bypass writes, but also allows writes to bypass writes. According to the original code (without the `membar` instructions), what output triples $(X, Y, Z)$ are possible with PSO but not with sequential consistency? Explain why.

**Problem 2.** (20 points) [CS&G 7.4, 7.7] In order to solve the following problems, you need to carefully study Example 7.1 (p. 460 of Culler, Singh, & Gupta) first.

(a) Reconsider Example 7.1 where the number of hops for an $n$-node configuration is $\sqrt{n}$. How does the average transfer time increase with the number of nodes? What about $3\sqrt{n}$?

(b) Reconsider Example 7.1 where the network is a simple ring. The average distance between two nodes on a ring of $n$ nodes is $\frac{n}{2}$. How does the average transfer time increase with the number of nodes? Assuming each link can be occupied by at most one transfer at a time, how many such transfers can take place simultaneously?

**Problem 3.** (25 points) In many cases with data-intensive algorithms, the data in the system needs to go through phases. Typical phases are the producing phase, the transforming phase, and the consumption phase. Suppose we have an MP system where the phases are distributed to different processors. Also, suppose we are looking at one particular set of data, and each processor could concurrently be dealing with many more data in different phases.
Assume that all the variables are shared and initialized as follows:

```java
define int A = 0;
define int B = 0;
define boolean produced = false;
define boolean transformed = false;
define boolean consumed = true;
```

The following program is given for three processors, one for each phase.

<table>
<thead>
<tr>
<th>Producer</th>
<th>Transformer</th>
<th>Consumer</th>
</tr>
</thead>
</table>
| 1. while (! consumed);
2. A = 1;
3. B = 1;
4. produced = true; | 1. while (! produced);
2. A = A + B;
3. B = 0;
4. transformed = true; | 1. while (!transformed);
2. print(A);
3. print(B);
4. consumed = true; |

(a) Show the output for one pass of these processes under the following memory models:

(i) SC – Sequential Consistency
(ii) PRAM – Pipelined RAM
(iii) PC – Processor Consistency

Explain the differences in output using the properties of the consistency models.

(b) Make necessary adjustments to the programs for each processor if RC (Release Consistency) is used. Make necessary assumptions and introduce variables (if needed).

(c) What would be the outcome if the RC model used in (b) were—

(i) ERC (Eager Release Consistency)
(ii) LRC (Lazy Release Consistency)?

**Problem 4. (20 points)** Calculate the number of physical “wires” needed between nodes for a 4096-node system of each of the following network types. For each network,

(i) Give a general expression for the number of wires needed to connect that type of network. Assume one “wire” for a connection between two nodes.

(ii) Give the diameter of the network.

(iii) Give the average distance between two nodes in this network.

(a) A hypercube network.

(b) A barrel-shifter network.

**Problem 5. (15 points)** Like the perfect shuffle, the butterfly permutation by itself does not give a completely connected network.

(a) Show how a butterfly interconnection partitions 4- and 8-node networks.

(b) Since the perfect shuffle did not provide a complete connection, the exchange was added to the network, as discussed on page 10 of Lecture 23. By looking at the bit-representations of the source and destination nodes, what “property” does the perfect shuffle permutation have which
prevents complete connection (as an obvious observation)? Does this property hold for the butterfly permutation as well, and if so, how? Will the introduction of an exchange permutation into the butterfly interconnection network provide a complete connection, as it did with the perfect shuffle? If so, prove it. If not, would it work for any special cases (e.g., networks of a particular size \( N \))?

(c) Would adding only the super-butterfly permutations be a way to provide a complete connection with a maximum path length of \( O(\log N) \)?