1. **Matching applications to architecture**

You've just finished a 10-week class in parallel processing, have graduated, and have been hired by a large San Diego based company that we'll call Q. During your orientation session, you learn that two computers are available to your project. Both rely on the same processor, but provide different numbers of processors, different amounts of memory, and a different interconnect. Each processor computes at a rate not to exceed 1 Gigaflop/sec., but typically in the range of 100 to 500 megaflops/sec.

   a. **UMA.net**—a Uniform Memory Access (UMA) multiprocessor with 64 processors and 32 GB of shared memory.

   b. **Distributed.net**—a loose connection of 145,000 processors with 9 Terabytes of RAM tied together over the internet with 56.6kbit/s network connections with an average latency of 800ms between processors. The 56k connections randomly go up and down.

Your project manager at Q gives you descriptions of two projects she's currently working on, and asks your opinion of what machine would be best for each program. You can assume that CPU time on all three machines has the same cost. The two projects are as follows:

**Project A [14 points]**: Due to their president's fondness for Japanese Poetry, Q Inc. has developed a new algorithm that analyzes haiku and gives it a score on how "good" it is. The algorithm is almost 100% serial—one can only find parallelism between instructions issued a few cycles apart from each other. Each poem takes 30 seconds on a single processor to compute. However, there is an enormous amount of haiku to process, and each poem can be processed separately from the other. Just the score of the haiku needs to be recorded in the results.

Discuss what sort of parallelism exists in this algorithm, and choose a machine that would run this algorithm best. Explain your decision. Would the other machine be more effective if only one Haiku needed to be analyzed? Why?

**Program B [10 points]**: Q Inc. has been contracted to design a new toll road for San Diego, and they need to perform a traffic flow analysis. Each intersection in the city is represented by a row in a matrix, and each road in the city a column. The analysis is performed by doing an LU decomposition on the data before and after the toll road is added. Since the matrix will be fairly large (approximately 70,000 x 70,000) Q wants to parallelize the process. Using your knowledge of LU decomposition, choose a machine that would run the simulation the fastest, and explain why.
2. **Shared memory programming**

a. Consider the following loop that runs in parallel on two processors. List all possible outputs that may result, explaining your answer. Assume that \( j \) and \( k \) are shared variables, and that \( i \) is private.

```plaintext
forall ( int i=0, j=0, k=0; i< 2; i++ )
    j = j + 10;
    k = j + 10;
}
cout << “k = “ << k << endl;
```

b. The following code performs barrier synchronization, where \( n_{proc} \) is the number of processors. Answer the following questions.

i) Explain how the code works, demonstrating correct operation on 3 processes.

ii) Why are two lock variables needed, \( arrival \) and \( departure \), and what must they be initialized to ensure correctness? Similarly, how must Count be initialized?

```plaintext
Void barrier( . . . )
{
    LOCK(arrival);
    Count++;
    If (Count < n_{proc}) UNLOCK(arrival);
    Else UNLOCK(departure);

    LOCK(departure);
    Count--;
    If (count > 0) UNLOCK(departure);
    Else UNLOCK(arrival);

    Return;
}
```
3. Data parallel performance

You are given the following two data parallel programs that perform the equivalent force computation on a system of \( n \) particles. Assume that the arrays \( F[], T[], \) and \( X[] \) are distributed BLOCKwise over processors, that they are ALIGNed with one another, and that \( n \) is much larger than the number of processors.

**Program 1:**

1. \( \text{for} \ (i=0; \ i<n; \ i++) \)
2. \( F += \text{abs}(X[i] - X); \)

**Program 2:**

1. \( T = X; \)
2. \( \text{for} \ (i=0; \ i<n; \ i++) \{ \)
3. \( F += \text{abs}(X-T); \)
4. \( T = \text{CSHIFT}(T,1); \quad \text{! circular shift} \)
5. \( \} \)

The \text{CSHIFT} circularly shifts the array \( T \) by one position to the left. Thus, if \( n \) were 4 and \( T \) contained \( \{5,7,0,6\} \), then executing \( T = \text{CSHIFT}(T,1) \) would result in \( T \) containing \( \{7,0,6,5\} \). Assume that a floating point operation executes in one time unit, that a scalar broadcast costs \( \alpha \log_2 P \) units, and that a \text{CSHIFT} entails sending a single-word point-to-point message at a cost of \( \alpha \) time units.

Which version of the program is faster? Be sure to justify your answer—showing all work—and discuss how you think the compiler parallelizes the code.

4. Partitioning.

You are given two 16-element arrays of data, \( x[16] \) and \( y[16] \). In the following two problems you will be asked to distribute these arrays over 4 processors in different ways and to comment on various aspects of performance.

1. Let both \( x[\ ] \) and \( y[\ ] \) be distributed in BLOCK fashion. How many array values are stored on processor 0, including ghost cells? How many are stored on processor 3? How many data values must be sent by processor 3 to execute the following loop correctly across the 4 processors?

\[
\text{forall} \ (i=0; \ i<15; \ i++)
\]

\[
y[i] = x[i] + x[i+1];
\]

2. Repeat question (1), assuming that the data distribution of both arrays is changed to CYCLIC.

3. Repeat question (1), assuming that the data distribution of \( x[\] \) is BLOCK and \( y[\] \) is CYCLIC.