There are four problems below. Please choose three to solve. If you choose to solve all four, only the lowest three scores will count. Show all your work and write in a readable way. If you think a question is ambiguous, do your best to make (and to state) any reasonable assumptions you need to derive a solution.

Question 1

Suppose you are given two sorted arrays, $A[1:n]$ and $B[1:m]$, distributed equally among P processors. That is, each processor holds n/P elements of A and m/P elements of B, with processor i holding the i^{th} consecutive block of elements from each array.

(a) Give a distributed memory algorithm to merge A and B, with the final result $C[1:(n+m)]$ (i.e., the merged $n+m$ elements) also equally distributed among processors.

(b) Derive the time complexity of your algorithm, as a function of m, n, and P. Assume that the time to send a message of size k words is $\alpha + \beta k$.

(c) What number of processes minimizes time, in terms of m and n?

For this question, you may assume m, n, and P are all powers of two, and that all elements of C are distinct. You may assume collective operations, like ALL2ALL, REDUCE, BROADCAST. To minimize ambiguity, clearly state the interface and functionality of any collectives you use.

Question 2

Consider the following recurrence, given the real-valued coefficients $a_1, \ldots, a_n$, $b_1, \ldots, b_n$, $c_1, \ldots, c_n$, and $d_1, \ldots, d_n$.

$$x_i \equiv \begin{cases} 0 & \text{if } i = 0 \\ \dfrac{a_i x_{i-1} + b_i}{c_i x_{i-1} + d_i} & \text{if } 1 \le i \le n \end{cases}. \tag{1}$$

(a) Give an efficient parallel algorithm for evaluating this recurrence, analyze it, and argue why it is efficient.

(b) A well-known method to estimate the square-root of a number, a, is to run n steps of the recurrence $x_i = \frac{1}{2}\left(x_{i-1} + (a/x_{i-1})\right)$, given some initial guess x_0. (This recurrence comes from Newton's method.) Suppose we wish to design a parallel algorithm for this problem using only simple addition, subtraction, multiplication, and division. Argue that any algorithm requires $\Omega(n)$ steps, i.e., that no speedup beyond a constant factor is possible.[1]

Question 3

Shared memory abstractions on a distributed memory platform.

You are designing a runtime library that will support the implementation of a shared memory programming model "reasonably" efficiently on a distributed memory platform. In about one to two pages, please answer the following questions.

(a) What are the most important characteristics of a shared memory programming model relevant to your task? Try to come up with at least three such characteristics.

[1]Parts (a) and (b) are part of a classical result about whether a recurrence given by a rational function, $x_i = f(x_{i-1})/g(x_{i-1})$, can be parallelized. When the maximum degree d of $f(x)$ and $g(x)$ is 1, as in part (a), the answer is "yes." But when $d > 1$, the answer is no. Well, assuming only primitive rational operations—allowing other operations, speedup may be possible.

(b) What are the major characteristics of a distributed memory platform relevant to your task? Try to come up with at least three such characteristics.

(c) Explain the challenges that your answers to (b) pose for the characteristics you identified in (a).

Question 4

Cache-efficient triangular solve.

Suppose we wish to solve the triangular system, $Lx = b$, for the dense vector, x, given an $n \times n$ lower triangular matrix, L, and dense right-hand side vector b. A conventional algorithm for such an operation might work as follows. First, logically partition L so that the system becomes:

$$\begin{pmatrix} \lambda_{11} & 0 \\ l_{21} & L_{22} \end{pmatrix} \cdot \begin{pmatrix} \chi_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \beta_1 \\ b_2 \end{pmatrix} \tag{2}$$

where λ_{11} is the scalar diagonal entry, l_{21} is a column vector, and L_{22} is the $(n-1) \times (n-1)$ lower trailing triangle. Next, solve for the first entry of x by computing $\chi_1 \leftarrow \lambda_{11}^{-1}\beta_1$. Then repeat this process on the smaller system, $L_{22}x_2 = b_2 - l_{21}\chi_1$, since L_{22} is also lower triangular.

(a) Suppose we execute the conventional algorithm on a sequential machine with a two-level memory hierarchy. That is, suppose the machine has an infinite slow memory in which all the data begins initially, as well as a fast but small processor-local memory that can hold at most Z words. Furthermore, suppose $n \gg Z$. Estimate the number of words this algorithm transfers between slow and fast memory. Justify your estimate.

(b) Give a more I/O-efficient algorithm, i.e., one that potentially moves fewer words of data between slow and fast memory. Analyze your algorithm.