1. Define a linear array of size $n$ with a bus to be a linear array augmented with a single global bus. Every processor is connected to the bus and in one unit of time, one processor can write to the bus and all other processors can read from it. This allows broadcasting in unit time, but only of one data word per time step.

(a) State an efficient algorithm to sum $n$ values, initially distributed one per processor, on such an architecture. What is the parallel cost of the algorithm? Compare to the case of finding the sum of $n$ values on a regular linear array without a bus.

(b) Can the parallel cost of the algorithm be improved by increasing the amount of data initially allocated to each processor?

2. Miller and Boxer, Chapter 5, Problem 1.

3. Miller and Boxer, Chapter 5, Problem 2.

4. Miller and Boxer, Chapter 5, Problem 3. In part (c), assume that each elements of the array to be sorted reside initially on a separate processor and that the goal is for the elements in the array to end up on separate processors again after the algorithm completes, but that they should be in order by processor ID.

5. In practice, many matrix computations involve sparse matrices. Data structure for storing sparse vectors and matrices store only the nonzero entries. Although this requires storing not only the values of the nonzeros, but their location within the vector/matrix, it is usually still more memory efficient to use such data structures than to explicitly store all matrix entries. This problem examines some of the details of the efficiency of these data structures.

(a) Describe in detail a data structure for storing a sparse matrix in C. Try to determine at approximately what level of sparsity (percentage) the sparse matrix data structure uses less memory than the explicit representation.

(b) Compare the efficiency with which such a sparse matrix can be initialized to all zeros with the efficiency with which the same operation can be performed using an explicitly stored matrix.

(c) Give a pseudocode implementation of a function for multiplying two matrices stored in your sparse representation without taking into account any cache effects.

(d) Analyze the efficiency with which two sparse matrices can be multiplied using your data structure and compare it to the efficiency with which the same operation can be performed on two explicitly stored matrices using a straightforward theoretical analysis that doesn’t take into account cache effects.
(e) In practice, how would you take cache effects into account in implementing sparse matrix multiplication? What are the differences (if any) from the case with explicitly stored matrices?