

Your source files must be dated on or before midnight, **Monday, 23 June 2008**.

OVERALL OBJECTIVE

You are to write a C or C++ concurrent *Shearsort* program using POSIX PTHREAD library and semaphore operations. You are NOT to use `fork()` calls, pipes, or message queues in this exercise.

The *Shearsort* is a simple mesh-sorting algorithm that consists of nothing more than alternately sorting rows and columns of the mesh. In particular, it sorts all the **rows** in Phases 1, 3, ..., $\log_2 \sqrt{N} + 1$, and all the **columns** in Phases 2, 4, ..., $\log_2 \sqrt{N}$, where N is the total number of elements. The **columns** are sorted so that smaller numbers move upward. The **odd rows** (1, 3, ..., $\sqrt{N} - 1$) are sorted so that smaller numbers move leftward, and the **even rows** (2, 4, ..., \sqrt{N}) are sorted in reverse order (*i.e.*, so that smaller numbers move rightward). The numbers will appear in a snakelike order after $2 \log_2 \sqrt{N} + 1 = \log_2 N + 1$ phases. An example is given in Figure 1. (A proof that the Shearsort algorithm works can be found in *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes* by F. Thomson Leighton. However, the information about Shearsort in this handout should be sufficient for you to complete the exercise.)

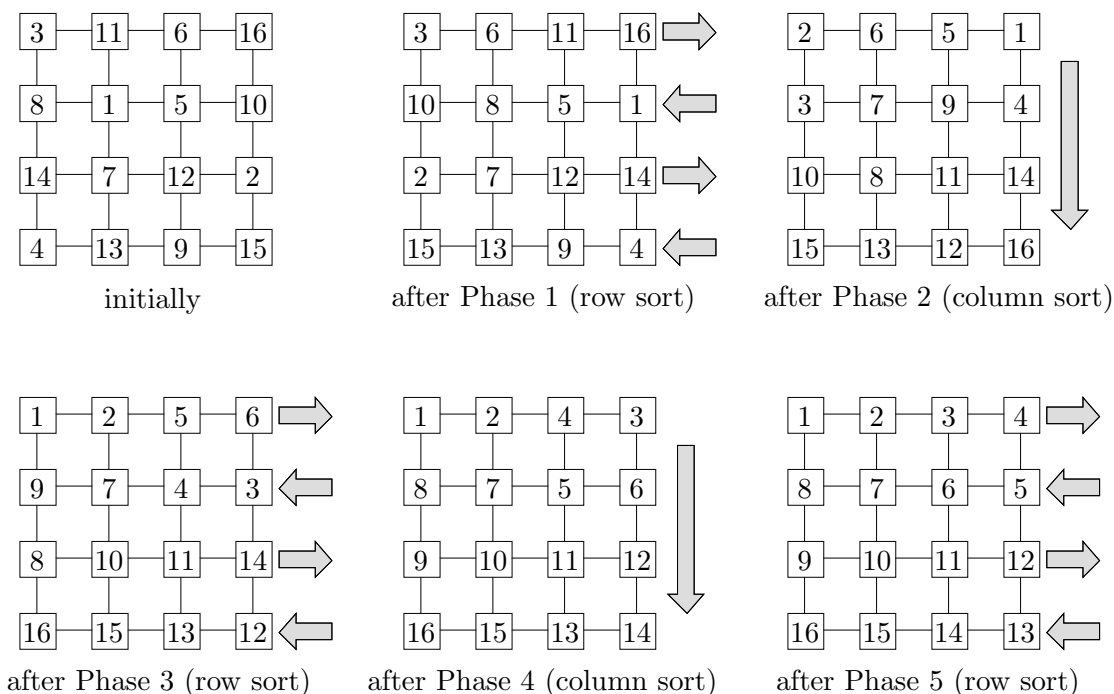


Figure 1. Alternately sorting the rows and columns in Shearsort. The numbers to be sorted appear in a snakelike order after $\log_2 N + 1$ phases. Notice that even rows are always sorted in reverse order. The arrow direction indicates the sorting order (from small to large).

The Shearsort algorithm will proceed similarly as described below. In this project assignment you will use shared memory to store ONE copy of the whole 2-dimensional array. Also, you will be required to create a limited number of threads for the sort. For a square array of n -by- n elements, ONLY n THREADS are to be created. Each thread is to alternate between row (odd) and column (even) phases. To synchronize the phases properly, you will need to use semaphores. You are to use at most ONE SEMAPHORE PER THREAD.

Your program should implement the Shearsort algorithm to handle 16 integers as follows.

1. Read the integers into a 2-dimensional $n \times n$ array in global memory from the file "input.txt". This can be done by the main program.
2. Print the integers in the order entered to stdout.
3. Create and initialize the semaphores necessary for the algorithm.
4. Create the n threads to sort the array using Shearsort.
5. Wait for the threads to finish.
6. Print the array of sorted integers to stdout.

Each thread would do the following in each phase of the algorithm.

1. By performing the appropriate number of wait operations, block until the prior phase (if any) is finished.
2. Sort the row/column in the appropriate order using Bubblesort.
3. Perform appropriate signal operations to signal the other threads to begin the next phase.

WARNING: You are NOT to use the sleep() call or any code (such as a loop counting from 1 to 10000) to introduce any delay in your program. If any such delaying code is found in your program, YOU WILL LOSE 30 POINTS.

ASSIGNMENT DELIVERABLES: The deliverables for this assignment consist of all C/C++ source code files which are part of your solution, as well as a make file which compiles and links your program. The makefile should be named Makefile. The executable program generated by the Makefile must be named proj02. The file containing your "main" function must be named proj02.driver.c. You are free to separate your program into multiple source files as you see fit.