Homework 4

Due as a typed, pdf file

1. (20 points) **MPI:** The prefix sum of a particular cell in a NxN matrix is the sum of its value and the values of all cells above and/or to the left of it. An example of a matrix and its prefix sum matrix is as follows:

Original Matrix

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Prefix Sum Matrix

1	3	6	10
6	14	24	36
15	33	54	78
28	60	96	136

We want to compute the prefix sum of any such square matrix in parallel using MPI. If we divide the NxN matrix amongst P^2 processes such that each process owns a (N/P)x(N/P) submatrix, the processes can be arranged in a PxP 2D topology to represent the original matrix. For example, we can divide the above 4x4 matrix amongst P^2 = 4 processes arranged in a 2x2 grid:

P0

1	2
5	6

P1

3	4
7	8

P2

9	10
13	14

P3

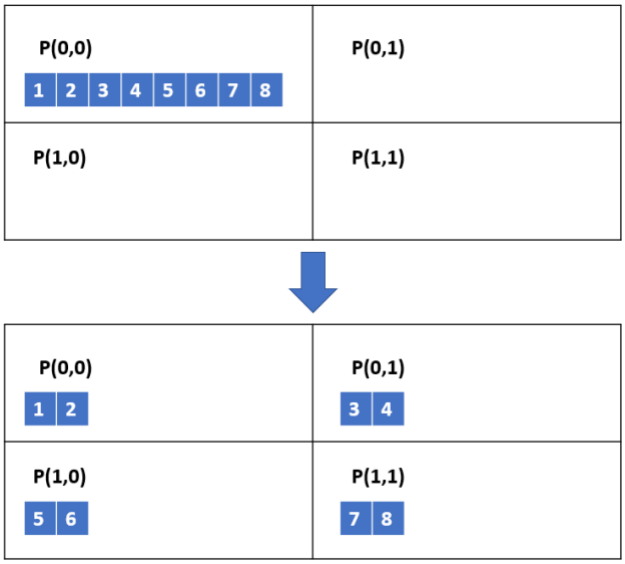
11	12
15	16

Given such an arrangement, write a MPI program to compute the prefix sum of the original matrix. You **do not** need to reconstruct the original matrix on a single process afterwards, it is enough to compute the submatrices only. In other words, it is okay to leave the above prefix sum matrix divided amongst the processes.

(Hint: arrange the processes in a 2D grid in MPI and use sub-communicators)

2. (15 points) **Cost Model:** Consider the following algorithm for the scatter (e.g. MPI_Scatter) operation, where NP words of data are being scattered from process 0 to each of the P processes, such that each process ends up with N words of data.

Organize the processors in a 2D grid, so each process has 2 coordinates (you may assume the number of processes P is a square). Process 0 with coords (0,0) sends the portion of the data that needs to be scattered among the processes in row i to process (i,0) via a point-to-point messages. Each process (i,0) in the first column then sends data that needs to go to each process in its row via individual point-to-point message. The following images illustrate the results of such a scatter for P=4 and N=2:



a. Write an expression for the completion time of this algorithm. Note (or assume) that you cannot start sending a second message until the first message has been sent out from your process.
b. Write an expression for the completion time of a simple algorithm where process 0 sends a separate message to each process with its N-word data.
c. Compare the two algorithms: under what condition(s) is the grid-based algorithm preferable to the simple algorithm?

3. (15 points) **One-sided MPI:** Consider P processes each with local arrays containing N `<int, val>` key-value pairs where the keys are integers from 0 to P-1 and the values are K-byte objects of type `val`. Write an MPI program that sorts all the elements in the processes such that process `i` contains only values whose key is `i` (i.e. P_0 has all values that had key 0, P_1 has all

values that had key 1, …, P_{P-1} has all values that had key P-1). Each process' final array should contain only `val` objects.

You may **not** assume that each key appears with the same total frequency. You must implement this using MPI_Get/Put instead of Send/Recv, but you may use collectives like MPI_Reduce/Scan/Gather/etc.