This assignment contains two problems. For Both we assume the following setup. Programs are run with
`mpirun -n` N (executable name)
where the number of processors N is a square, say $N = p^2$.

In both problems we refer to a $n \times n$ matrix M. We assume M to be logically divided into p^2 square submatrises, each of size $(n/p) \times (n/p)$.

We denote the (i, j)-th processor by $P_{i,j}$ and (i, j)-th submatrix of M by $M_{i,j}$ where $0 \leq i, j < p$. Initially $P_{i,j}$ should generate $M_{i,j}$, so that at the begining of the given algorithm outline we assume that $M_{i,j}$ is already available to $P_{i,j}$

Cannon's Matrix Multiplication

Notation: For an integer x, $\bar{x}$ is the integer s.t. $0 \leq \bar{x} < p$ and $x \equiv \bar{x} \mod p$

Algorithm outline

Goal: Given $n \times n$ matrices A and B compute $C = AB$.

Initially $P_{i,j}$ generates $A_{i,j}$ and $B_{i,j}$

Perform cyclic shift on i-th row of A by i places so that $P_{i,j}$ contains $A_{i,\overline{i+j}}$
Perform cyclic shift on j-th column of B by j places so that $P_{i,j}$ contains $B_{\overline{i+j},j}$

In $P_{i,j}$, compute $C_{i,j} = A_{i,\overline{i+j}} B_{\overline{i+j},j}$

For $k = 1, \ldots, p-1$ do the following
(Notice that at the begining of k-th iteration, $P_{i,j}$ contains $A_{i,\overline{i+j+k-1}}$ and $B_{\overline{i+j+k-1},j}$)
Perform cyclic shift on all rows of A by 1 place so that $P_{i,j}$ contains $A_{i,\overline{i+j+k}}$
Perform cyclic shift on all columns of B by 1 place so that $P_{i,j}$ contains $B_{\overline{i+j+k},j}$
In $P_{i,j}$, compute $C_{i,j} = C_{i,j} + A_{i,\overline{i+j+k}} B_{\overline{i+j+k},j}$

(Notice that at the end of $p-1$-th iteration, $P_{i,j}$ contains $C_{i,j}$ is indeed the submatrix corresponding to $C = AB$)

Matrix Vector Multiplication

Goal: Given $n \times n$ matrice A and n-dimensional vector x compute $y = Ax$. We logically partition x into p many n/p dimensional vectors $x_0, \ldots, x_{p-1}$.

Initially $P_{i,j}$ generates $A_{i,j}$ and $P_{0,j}$ generates x_j
$P_{0,j}$ broadcasts x_j to $P_{i,j}$ for all i
$P_{i,j}$ computes $y_{i,j} = A_{i,j}x_j$
$y_{i,j}$'s are reduced to $P_{i,i}$ as $y_i = \sum_j y_{i,j}$
$P_{i,i}$ sends y_i to $P_{0,i}$
(Notice that y_i's are indeed the subvectors of $y = Ax$)