CS594
Yicheng Li

1. The ring application will have token passing around among the ranks. The first rank only does send. Ranks in the middle will do receive first and do send to the next rank. The last rank will do only receive. This will prevent ranks from blocking each other.

2 & 3. The ranks will be split into multiple communicators. Each row and each col will have a communicator. The number of communicators = number of rows + number of cols.
Each communicator owns a token. And the token will be passed from the first to the last rank in communicator.
To avoid blocking, I first let tokens passing around in col communicators and then let tokens passing around in row communicators.

4. Matrix transpose.
The user will specify a NxN matrix with N as input, PxQ processors where P=Q as assumption. Thus, N, P, Q are the three input for the program. The number of ranks will also be an input.
The program will first initialize a NxN matrix in global space, for testing propose. The matrix is stored in one-dimensional array. And since each process is assigned to a block, row index and col index is calculated for assignments. Then the program will calculate the pointer of first element of that block and pass this pointer to a function, along with P, Q and N. The function will first pack the data of the block into a contiguous buffer. Do transpose on the block and it is ready for send.
Doing transpose within each block and switch blocks will result in the same answer as doing transpose for the whole matrix.

After doing transpose for each block, the upper triangle will send its block to corresponding lower triangle and then receive from lower triangle. The diagonal only needs to do a transpose on its own. But for training propose, it will send the block to itself.
The corresponding block is calculated by switching row and col numbers.
The final result is only viewable locally for each rank.

Note:
The program has a strict rule for inputs.
The number of ranks has to equal to P * Q.
P must be equal to Q.
N has to be divisible by P and Q.