Name: Drew Masters
Date: 9/23/2016
Class: COSC 462

Homework 2

Problem one, as I understood it, was to take a grid of processors and create a communicator along the diagonals. This was done by taking the two extreme process that would be their own diagonals create a communicator that contain just that process. I did this with a if statement and a else if statement for the positions that would have these single process diagonals. The next step was to find the middle diagonals and to break them up if need be. The three center diagonals (the actual diagonal of the grid and the diagonals on either side of it) did not require any additional processing except for the mod operation to break them into diagonals.

The diagonals between these center diagonals and the single diagonals required being split so the diagonal did not cross the grid. If you imagine the grid as a upper right triangle and a lower left triangle, these diagonals would wrap around the main diagonal from one of these triangles to the other one. I found which side of the grid the process belonged to and split the diagonal such that all of the processes in one triangle were in the same communicator.

Problem two involved passing a message down the resulting communicator and back to the starting position on the communicator. For example the message would start at the process that has rank zero on that communicator and pass it down the communicator to the next rank on the communicator until it reached the last process in the communicator (the process with maximum rank). Once the message reaches this process it is passed back directly to the starting process and the message passing for that communicator ends.

Problem three was to divide the process grid into communicators by diagonals and antidiagonals. The antidiagonal are the diagonals of the mirrored grid. The same process used to create the communicators in problem one was applied to the diagonals and the antidiagonals. The logic of the if statements and the else if statements was changed to get the correct processes for the single process communicators. The mod operation was also modified to get the correct processes but the split operation remained the same.

Problem four was to pass a token starting at the process that had rank zero in the communicator until it reached the starting process again. The token was passed alternating the communicators, from antidiagonal to diagonal or diagonal to antidiagonal. Once the token gets back to the starting process and the next communicator it was going to be passed to is the same as the communicator it started on the token is finished. The process lets all of the other processes know that the token is finished. Once all tokens are finished all processes end.

Issues

The code does not terminate correctly after all of the tokens are finished being passed in problem 4. The program has to be killed. The tokens are being passed around the communicators correctly. The problem is that the processes are terminating until only one process is left and it does not end.

Steps to run code

-Run make. It will generate HW2_1 and HW2_2. HW2_1 are problems 1 and 2, HW2_2 are problems 3 and 4. You can run make clean to remove the executables.

-The code will only run if the number of processes are a perfect square root i.e. 4, 9, 16, 25, or 36. Otherwise it will terminate and give a usage message.

mpirun -np number_of_processes HW2_x

(if running HW2_2 will have to kill program)