## CSC 410 Final Exam – Part 2

Guidelines: the work should be your own.

Demonstrate your understanding & mastery of the material.

*Partial programming & documentation : discuss algorithms & code fragments here. You* ***don’t*** *need to provide the full program, just show the important & interesting parts. Convince me you know what is going on.*

10) MPI-Warm-Turnip … sort of like Hot Potato but not nearly as exciting! **( 15 pt )**

Again, comment, explain, provide pseudo code fragments, but you are not required or expected to fully implement the program. Assume the MPI work has been set-up and there are now P processes with IDs 0, 1,2,3…. They are all ready for your direction. Explain now how you would implement the Warm-Turnip behavior:

* Exactly 1 process can have the turnip at a time
* The process with the turnip will pass the turnip to some other process [[ randomly selected ]]
  + Can’t send back to the one that sent it to you, unless only 2 nodes in the world
* That process will accept the turnip, print a message stating it has the turnip, and then pass it along
* This continues indefinitely (termination is left to another developer to worry about)

Explain:

I start out by initializing the send from node 0. To do this I get a random number between 1 and the world size. I then send it to the random node using MPI\_Send. This happens after I say, “HOT TURNIP!!!” This way everyone knows the game has started. The node that the turnip is sent will then receive the turnip along with the status of the previous sender. After this the node will declare they have the turnip. The node will then find a random node to send the turnip to next between 1 and the world size. It will calculate a new node if the random number is itself or the previous node. If the world size is three and the random number is the previous node it will allow for the node to be used. The reason it is 3 instead of 2 is because node 0 is only being used to initiate the game and does not participate in hot turnip. Once the node is figured the node will say who they are sending it to, change the turnip to zero because they don’t have it anymore, and wait at MPI\_Recv until it has had the turnip sent to it again.

Code:

**int** turnip = 0;

    MPI\_Status status;

    if(wrank == 0) {

**int** node = (rand() % (wsize - 1)) + 1;

        turnip = 1;

        printf("HOT TURNIP!!!\n");

        MPI\_Send(&turnip, 1, MPI\_INT, node, 0, MPI\_COMM\_WORLD);

    } else {

*// while(game continues) {*

        MPI\_Recv(&turnip, 1, MPI\_INT, MPI\_ANY\_SOURCE, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &status);

        printf("%d has the turnip\n", wrank);

        usleep(100000);

**int** node = (rand() % (wsize - 1)) + 1;

        while((node == status.MPI\_SOURCE && wsize != 3) || node == wrank) {

            node = (rand() % (wsize - 1)) + 1;

        }

        printf("Sending to %d\n", node);

        MPI\_Send(&turnip, 1, MPI\_INT, node, 0, MPI\_COMM\_WORLD);

        turnip = 0;

*// }*

    }