ECE 473 --Assignment 02
Due Date Specified in BlackBoard
Ring of Processes

Imagine that all processes are logically connected in a ring topology as follows: Therefore, processes (not processors) can send and receive data only to and from their adjacent neighbors. Note that the ring wraps around, i.e. process 0's left neighbor is process (n-1) and process (n-1)'s right neighbor is process 0.

Each process has a value “myNum” that is a randomly generated integer number between 0 and 100. Be sure each process seeds its random number generator with its rank.

Each process will first print it's own ID, it's myNum value, as well as the ranks of it's left and right neighbor, all on a single line. Then each process will exchange myNum values with its neighbors. Each process will then print its own IP, and the values it received from its neighbors, all in a single line.

A sample execution of this program on 6 processes should look like this, (with the possible exception of the random number values and the order the prints occur in).

Process [1] has myNum = 33, R_rank = 2, L_rank = 0
Process [0] has myNum = 97, R_rank = 1, L_rank = 5
Process [5] has myNum = 26, R_rank = 0, L_rank = 4
Process [4] has myNum = 40, R_rank = 5, L_rank = 3
Process [3] has myNum = 4, R_rank = 4, L_rank = 2
Process [2] has myNum = 69, R_rank = 3, L_rank = 1
Process [5]: Recv'd from right = 97, Recv'd from left = 40
Process [0]: Recv'd from right = 33, Recv'd from left = 26
Process [3]: Recv'd from right = 40, Recv'd from left = 69
Process [4]: Recv'd from right = 26, Recv'd from left = 4
Process [1]: Recv'd from right = 69, Recv'd from left = 97
Process [2]: Recv'd from right = 4, Recv'd from left = 33

In terms of MPI calls, you may only make use of MPI_Send() and MPI_Recv(), no other point-to-point or collective communication constructs may be used. Your solution must be guaranteed deadlock free, regardless of any internal MPI buffering. Communication must also be done in such a way that multiple processes are communicating at the same time. If you think you need a loop to carry out the communication, you're doing it the wrong way.

Chapter 3 of the MPI Specification Text is a valuable repository of knowledge and examples. In particular, section 3.5.
Your project will have the following files:

```
Makefile
first_last_ass2_ring.c
```

These files will reside in a directory called

```
HW02_first_last
```

where first and last are your first and last names, respectively. The Makefile will be a properly formatted Makefile with both an 'all' and a 'clean' section.

Test your code on 1, 2, 4, 8, and 16 processors both on your local machine, as well as on Palmetto. Make sure you name your jobs accordingly in the pbs script so that I will be able to look at the .e and .o files and tell them apart. Make sure all files reside in your directory.

To submit your project, you must tar gzip your project directory, and it's contents by:

```
tar cf - ./HW02_first_last | gzip > HW02_first_last.tar.gz
```

You will perform this operation from the parent directory of

```
HW02_first_last
```

Submit your tar.gz files on BlackBoard by the time and date indicated on the BlackBoard site. No late work will be accepted.