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:
Process rank is connected only to processes

(rank + 1) % size and

(rank + SIZE – 1) % size.

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 (size - 1) and process (size - 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 exchange myNum values with its two neighbors. Each process will then print its own rank and its original number, then print the rank of its two neighbors and the sum of its number and numbers sent to it by its neighbors, all in a single line.

A sample execution of this output follows, note that the Sum for 5 equals myNum for 4, 5, and 6 added.

Rank [5] has myNum = 26, R_rank = 4, L_rank = 6, Sum = 70
Rank [4] has myNum = 40, R_rank = 3, L_rank = 5, Sum = XX
Rank [6] has myNum = 4, R_rank = 5, L_rank = 7, Sum = YY

Now, make a copy of this program and modify it as follows: This program should work exactly like the other one, except now tasks are able to communicate with several other tasks as follows:

Process rank is able to send to

rank + 2^0 mod size

rank + 2^1 mod size

rank + 2^2 mod size

rank + 2^{k-1} mode size

Where k satisfies SIZE >= 2^k

Similarly, each process is able to receive from

rank + size - 2^0 mod size

rank + size - 2^1 mod size

rank + size - 2^2 mod size

rank + size - 2^{k-1} mode size

Now the program goes though k rounds, such that in each round r
sending one message to rank + 2^r and
receiving another from rank + size – 2^r

In each round first send the partial sum, and then receive and add the partial sum provided.

As an example, if SIZE == 8, k == 3, so task 4 will execute 3 rounds:

Send sum to 5, receive delta from 3, then add delta to sum
Send sum to 6, receive delta from 2, then add delta to sum
Send sum to 0, receive delta from 0, then add delta to sum
Print sum

In terms of MPI calls, you may only make use of MPI_Send(), MPI_Recv(). And MPI_Sendrecv, 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.

Your project will have the following files:

```
Makefile
first_last_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 output files including the number of processors 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.