ECE 473 --Assignment 0
Due Date Specified on Canvas
Small Ball On Palmetto

Log in to Palmetto (user.palmetto.clemson.edu) and create a file called "smallball.c" with the a program that has each MPI task pass a "ball" one at a time to the next task in numerical order and then wail for the previous task to pass the ball and repeat until the program ends. The program must make sure the passing is synchronous, which is to say no task completes sending until the receiver of that send has completed receiving the ball. This can be accomplished by immediately following each send with a recv. And each recv with a send. For example:

Send:

MPI_Send(p, c, d, r+1, t, MPI_COMM_WORLD);
MPI_Recv(q, c, d, r-1, t, MPI_COMM_WORLD, &status);

Recv

MPI_Recv(p, c, d, r-1, t, MPI_COMM_WORLD, &status);
MPI_Send(q, c, d, r+1, t, MPI_COMM_WORLD);

Or at least something like that. Sine there is only one ball, starting at task zero, the you start with task 0 doing a send and all of the other tasks doing a recv. Whenever a task completely a send or a recv it flips and does the reverse, so the ball starts at 0, moves to 1, then 2, 3, 4, etc. and finally back to 0. Do not confuse the send/receive for synch shown above with alternating send/recive for the ball.

```
/* C Example */
#include <stdio.h>
#include <mpi.h>
```

At the command line, compile smallball.c and run it with 4 processes like this:

```
mpicc -g -Wall -Wstrict-prototypes -o smallball ./smallball.c
mpiexec -n 4 ./smallball
```

-g means turn on debugging, -Wall turns on most warnings, and -Wstrict-prototypes will enable further checking you will likely need during the course. If you have installed MPI on your laptop, you can do everything up to here on your laptop as well.

Note that mpirun, mpiexec, and orterun are all exactly the same thing.
As an argument to any of these, the following: -c, -n, --n, -np <#> are all exactly the same.

Now create a new file named "smallball.pbs" with the following contents:

```
#!/bin/bash
#PBS -N smallball
```

```
#PBS -l select=2:ncpus=8:mem=2gb:interconnect=mx
#PBS -l walltime=00:10:00
#PBS -j oe

source /etc/profile.d/modules.sh
module purge
module add gcc/4.8.1 openmpi/1.8.1

NCORES=`qstat -xf $PBS_JOBID|grep List.ncpus|sed 's/^.\{26\}//'`

cd $PBS_O_WORKDIR

mpiexec -n $NCORES ./smallball
```

Make sure the NCORES line comes out all on one line – and lone lose the closing back quote at the end. Notice this line has both regular single quotes (‘) and back quotes (`)

Now you need to set the environment variable PBS_O_WORKDIR to the directory where your program is:

export PBS_O_WORKDIR=/home/yourid/subdir_if_any

And finally you are ready to run the program

qsub smallball.pbs

When the program is done you should have files smallball.oXXXXXXX and/or smallball.eXXXXXXX show up in your working directory (the one you set above). If there is an “o” file, that is your output, and it should look similar to what you go above. If there is an “e” file, that is where errors are printed, and you probably did something wrong. It might take a few minutes before these files appear. You can use the qstat command to check on the job.

You should modify your program to print out useful things as it runs using printf as allows. The time spend running the program is always useful.

Turn in via Canvas your .c and .h files, .o and/or .e files and your .pbs file Also turn in a README in text that describes any special features or information needed to compile and run the program.

Appendix:

Check out how manual (man) pages work. Man pages are very useful when trying to understand what a function does and what parameters it takes. Type this at the commandline:

man MPI_Init

and you should see something like:

```
MPI_Init(3)                                              MPI

NAME
     MPI_Init - Initialize the MPI execution environment
SYNOPSIS
     #include "mpi.h"
     int MPI_Init(int *argc, char ***argv)
INPUT PARAMETERS
     argc - Pointer to the number of arguments
     argv - Pointer to the argument vector
…
```