**CSC 338 Parallel and Distributed Programming**

**Assignment 2: MPI Programming** **Due April 5, 2017**

**Background**

Monte Carlo methods solve problems through simulating a random process. One well known Monte Carlo algorithm estimates the value of *pi* by simulating tosses at a dartboard. Imagine that you have a square dartboard with sides of length 2 feet. There’s a circle on the dartboard with a radius of 1 foot—so its area is *pi* square feet. Darts tossed always hit the board and are uniformly distributed. Because the ratio of the area of the circle to the area of the square is pi / 4, the number of darts that land inside the circle approximately satisfy the equation:

Number in circle / total number of tosses = pi / 4.

**Procedure**

Your task is to estimate the value of pi by carrying out this Monte Carlo simulation. The algorithm is:

number\_in\_circle = 0;

for (toss = 0); toss < number\_of\_tosses; toss++) {

x = random double between -1 and 1;

y = random double between -1 and 1;

distance\_squared = x \* x + y \* y;

if (distance\_squared <= 1) number\_in\_circle++;

pi\_estimate = 4 \* number\_in\_circle / ((double) number\_of\_tosses);

Write a program to implement this algorithm using MPI. Process 0 should read in the total number of tosses and broadcast it to the other processes. Use MPI\_Reduce to find the global sum of the local variable number\_in\_circle and have process 0 print the result. You may want to use long long ints for the number of hits in the circle and the number of tosses because both may have to be very large to get a reasonable estimate of pi.

Try your program with several different values for number of tosses and with several numbers of processes. Write a short report using a table to show how the estimate changes with number of tosses. Also report how the number of processes affects execution time; I have not asked you to make formal timing measurements so this should be an informal assessment—do you feel that adding more processes shortens or increases the execution time, or has no effect? You don’t have to try many different process numbers—perhaps 2, 4, 8, and 12 processes, and with only one number of tosses that gives a reasonable estimate of pi.

Also, I haven’t specified how you get the number of tosses. You may get it from the command line or by prompting the user and using scanf(). Document your method in a comment at the top of the program.

**Deliverables**

Put your program and report in a folder named Asn2 in your upload folder on trace by five minutes after class time on the due date.