1 MPI Hands-On; Collective Communications II

1. **Different datatypes with a single MPI broadcast**, A program code14.c that broadcast routine is used to communicate different datatypes with a single MPI broadcast (**MPI_Bcast**) call.
   - MPI datatypes are used.
   - All processes exit when a negative integer is read.
2. **A SPMD program using broadcast and non-blocking receive.** The program consists of one sender process and up to 7 receiver processes.
   - The sender process broadcasts a message containing its identifier to all the other processes.
   - They receive the message and send an answer back, containing the hostname of the machine on which the process is running.
   - The receiving process waits for the first reply with MPI_Waitany, and accepts messages in the order they are received.
3. **A SPMD program that uses MPI_Scatter.** The program should be run with an even number of processes.
   - Process zero initializes an array of integers x,
   - then distributes the array evenly among all processes using MPI_Scatter.
4. **A SPMD program that uses MPI_Gather.** The program should be run with an even number of processes.
   - Each process initializes an array x of integers.
   - These arrays are collected to process zero using MPI_Gather and placed in an array y.
5. **Timing comparison of processes and thread creation**. Comparing timing results for the **fork()** subroutine and the **pthreads_create()** subroutine. code39.c, code40.c
   - Timings reflect 50,000 process/thread creations, were performed with the *time* utility (units are in seconds). Execute as

     ```
     time -p code39
     time -p code40
     ```

1.1 HOMEWORK - Due to December 13, 2010

1. **Computation of PI number**. This example (code15.c) evaluates π by numerically evaluating the integral

$$\int_0^1 \frac{1}{1+x^2}dx = \frac{\pi}{4}$$

   - The master process reads number of intervals from standard input, this number is then broadcast to the pool of processes.
   - Having received the number of intervals, each process evaluates the total area of **n/pool_size** rectangles under the curve
   - The contributions to the total area under the curve are collected from participating processes by the master process, which at the same time adds them up, and prints the result on standard output.
   - This code computes PI (with a very simple method) but does not use **MPI_Send** and **MPI_Recv**. Instead, it uses *collective* operations to send data to and from all of the running processes.
     - The routine **MPI_Bcast** sends data from one process to all others.
     - The routine **MPI_Reduce** combines data from all processes (by adding them in this case), and returning the result to a single process.
   - Run program for PI.
   - Rewrite this code that replace the calls to **MPI_Bcast** and **MPI_Reduce** with **MPI_Send** and **MPI_Recv**.

2. **Finding prime numbers**. Write a complete program that finds the prime numbers up to 10000.

   - The **FIRST** node should distribute the numbers to **ALL BUT THE LAST** node.
   - The nodes which receive numbers should pass the number to the **LAST** node, which is continuously waiting for primes.
   - When a node finishes its given numbers, it should pass `"-1"` to the **LAST** node.
   - The program should end when the **LAST** node receives `"-1"` from all calculating nodes.