1 MPI Hands-On; Collective Communications I

1. **Broadcasting an integer value to all of the MPI processes**, A program code13.c that reads an integer value from the terminal and distributes the value to all of the MPI processes.

   - Each process should print out its rank and the value it received. Values should be read until a negative integer is given as input.
   - You may find it helpful to include a **fflush(stdout)** to the code; after the **printf** calls in your program. Without this, output may not appear when you expect it.

2. **Broadcasting the name of the master process**, A program (code12.c) that first broadcasts the name of the master process then each nodes send hello messages to master node.

3. **Exercise: Computation of PI number as an example**. 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**.