# Assignment 1 - Point to Point communications

Start Assignment

- Due No due date
- Points 20
- Submitting a text entry box

## Exercise 1: Hello world

- Write an MPI program which prints the message "Hello World"
- Modify your program so that each process prints out both its rank and the total number of processes P that the code is running on, i.e. the size of `MPI_COMM_WORLD`.
- Modify your program so that only a single controller process (e.g. rank 0) prints out a message (very useful when you run with hundreds of processes).
- What happens if you omit the final MPI procedure call in your program?

## Exercise 2: Sharing Data

- Create a program that obtains an integer input from the terminal and distributes it to all the MPI processes. Each process must display its rank and the received value. Keep reading values until a negative integer is entered.

**Output Example**

**10**

**Process 0 got 10**

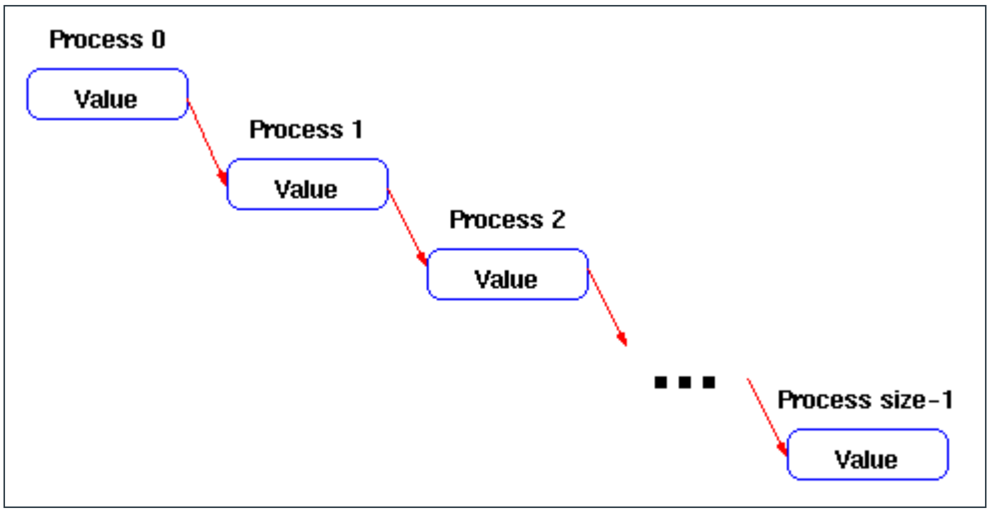
**Process 1 got 10**

## Exercise 3: Ping-Pong

- Implement a ping-pong program using MPI. Process 0 sends an initial message (an integer) to process 1, which then increments the value and sends it back to process 0. This exchange happens for a fixed number of iterations. Track and print the value of the integer at each step along with the sender and receiver's rank.

## Exercise 4: Sending in a ring (Broadcast by ring)

- Write a program that takes data from process zero and sends it to all of the other processes by sending it in a ring. That is, process i should receive the data add the rank of the process to it then send it to process i+1, until the last process is reached. Assume that the data consists of a single integer. Process zero reads the data from the user. print the process rank and the value received.



## Exercise 5: Solving 1D advection equation

- Implement a parallel solver for the 1D advection equation using the function above:

```
def solve_1d_linearconv(u, un, nt, nx, dt, dx, c):

    for n in range(nt):
        for i in range(nx): un[i] = u[i]
        for i in range(1, nx):
            u[i] = un[i] - c * dt / dx * (un[i] - un[i-1])
    return 0
```