

## Assignment 1 – Milestone

**Part 1:** Compile and run the program to replicate the output given in the textbook. First with one process, then with 4.

```
a1810750@pdclogin:~$ mpicc -g -Wall -o mpi_hello hello.c
a1810750@pdclogin:~$ mpiexec -n 1 ./mpi_hello
Greetings from process 0 of 1!
```

Screenshot of hello.c running with 1 process

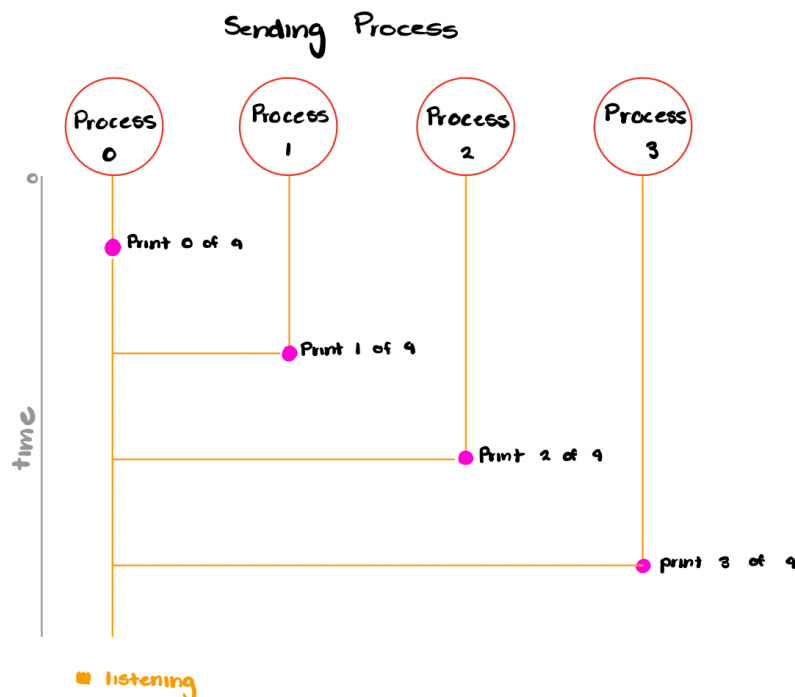
```
a1810750@pdclogin:~$ mpiexec -n 4 ./mpi_hello
Greetings from process 0 of 4!
Greetings from process 1 of 4!
Greetings from process 2 of 4!
Greetings from process 3 of 4!
```

Screenshot of hello.c running with 4 processes

**Part 2:** Write a brief and precise explanation of how the program works.

This program prints a message saying hello from each of the processes run. The program has two main functions. The first function sends the message from one process to another, whilst the other function receives the message and displays the greeting to the user.

**Part 3:** Draw a diagram showing all the messages sent when executed with 4 tasks. Number the messages in the order in which they occur.



**Part 4:** Will the messages be ordered in the same way for all executions of the program? Explain.

For all executions of the program, the message will be ordered the same. The program contains a for loop which increments from 0 to the number of specified processes. The for loop contains the receive function (MPI\_Recv) which passes through the current process, meaning that the

receives will only ever be passed in order from other processes. Overall, this implies that for all executions of the program the messages will be ordered the same.

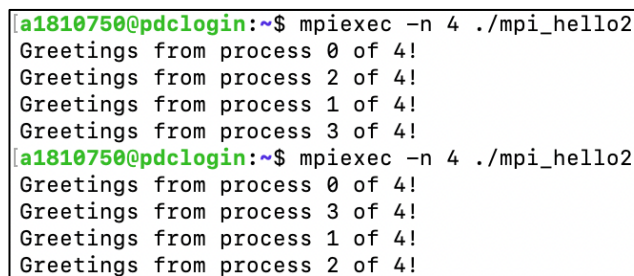
**Part 5:** Create a second version of the program with only one change. This change is to use the special constant `MPI_ANY_SOURCE` in the call to receive.

**Part 6:** Explain what effect, if any, you expect this change to have.

By changing `q` to `MPI_ANY_SOURCE` in the `MPI_Recv` function will mean that the process will no longer return in order. Instead, the processes will be returned in random order because the receive function will be listening to all sources, not just the one being specified.

**Part 7:** Then try to observe whether or not this happens. Explain your method and present the results.

I changed the `MPI_Recv` function to take in `MPI_ANY_SOURCE` instead of `q`. In order to check if the messages are being received in a random order, I ran the program twice. As seen below, the messages are not being received in order and each time the order of messages changes. This randomisation would be more evident with a greater number of processes.



```
[a1810750@pdclogin:~]$ mpiexec -n 4 ./mpi_hello2
Greetings from process 0 of 4!
Greetings from process 2 of 4!
Greetings from process 1 of 4!
Greetings from process 3 of 4!
[a1810750@pdclogin:~]$ mpiexec -n 4 ./mpi_hello2
Greetings from process 0 of 4!
Greetings from process 3 of 4!
Greetings from process 1 of 4!
Greetings from process 2 of 4!
```

Screenshot of program taking in `MPI_ANY_SOURCE`.