Distributed Memory Programming with MPI
Programming Assignment – 2

Exercise – 1 [2 points]

Write an MPI program where each of *p* processes, generates 20 random integers in the range]-100, 100[. Then, process 0 must print the single largest integer generated and the rank(s) of the process(es) containing it. Use of `MPI_MAXLOC` is not allowed. Run examples with p>=8 to demonstrate the correct implementation of your code.

Exercise – 2 [4 points]

Write your own implementation of `MPI_Bcast` using `MPI_Send` and `MPI_Recv`. Your routine, `myMPI_Bcast`, should take the same arguments as the original MPI_Bcast; that is :

```
int myMPI_Bcast(void *buffer, int count, MPI_Datatype datatype, int root,
                                MPI_Comm comm);
```

Subsequently, write an MPI program where the root process initializes a vector of three elements with random values and uses your routine to broadcast the contents of the vector to the other processes. Identify and run appropriate examples to demonstrate the correct implementation of your code.

Exercise – 3 [4 points]

Write your own implementation of `MPI_Reduce` using `MPI_Send` and `MPI_Recv`. Your routine, `myMPI_Reduce`, will have the following signature:

```
int myMPI_Reduce(const void *sendbuf, void *recvbuf, int count, int root,
                                MPI_Comm comm);
```

This should be a simplified version of the original `MPI_Reduce` which assumes double precision data (double) and performs the equivalent to the reduction operation `MPI_MIN`.

Subsequently, write an MPI program where each process generates one random value and uses your routine to obtain the minimum among them.