Ex06_p03

- As one would expect with periodic boundary conditions each process sends and receives a message. As one can see from the output of p03.cpp the sender and receiver process ids wrap around.

p03.cpp
```
#include <mpi.h>
#include <iostream>

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int message[1];

    // Always send the message, but to (rank + 1) % size
    message[0] = rank;
    MPI_Send(message, 1, MPI_INT, (rank + 1) % size, 0,
MPI_COMM_WORLD);
    std::cout << "Process " << rank << " sent message to
process " << (rank + 1) % size << std::endl;

    if (rank != 0) {
        MPI_Status status;
        MPI_Recv(message, 1, MPI_INT, rank - 1, 0,
MPI_COMM_WORLD, &status);
        std::cout << "Process " << rank << " received
message from process " << message[0] << std::endl;
    }

    // Process 0 receives the message from the last process
    if (rank == 0) {
        MPI_Status status;
        MPI_Recv(message, 1, MPI_INT, size - 1, 0,
MPI_COMM_WORLD, &status);
        std::cout << "Process " << rank << " received
message from process " << message[0] << std::endl;
    }

    MPI_Finalize();
    return 0;
}
```

Output from p03.cpp

```
Process 0 sent message to process 1
Process 0 received message from process 3
Process 1 sent message to process 2
Process 1 received message from process 0
Process 2 sent message to process 3
Process 2 received message from process 1
Process 3 sent message to process 0
Process 3 received message from process 2
```