Parallel and Distributed Programming
Laboratory 5

Steps:

1. Create a working directory (eg. Lab 5).

2. Compile (mpicc) and run (mpirun) a sample program.

   LINUX:

   ```
   mpicc example.c -o ex_mpi
   mpirun -np 7 ./ex_mpi
   ```

   WINDOWS:

   ```
   mpiexec -n 7 ex_mpi.exe
   ```

3. Write a program that for a given number, propagates it in the convention of the ring. Process number *i* should receive entered value from the process number *i-1* and send it further to *i+1* until the last process is not reached. The values should be read-out until the entered value is not negative.

   A sample output:

   ```
   Enter a number:
   5
   Process 0 got a 5
   Process 1 got 5 from process 0
   Process 2 got 5 from process 1
   ...
   Process 20 got 5 from process 19
   Process 0 got 5 from process 20
   Enter a number:
   …
   ```

4. Modify previous program with the use of the non-blocking versions of the send and receive procedures.

Configuration information:

There are several implementations of MPI, and it requires compilation and running tools:

- On Linux, there are two free MPI implementations - mpiCH and OpenMPI - to support the latter, you need to install the openmpi-bin and libopenmpi-dev packages for Debian based systems - (e.g. Ubuntu) or openmpi and openmpi-devel for systems based on Ret Hat distribution (e.g. Fedora)
- There is a version of Microsoft MPI in Windows - most often installed as a bundle with Visual Studio, but also available for self-installation https://docs.microsoft.com/en-us/message-passing-interface/microsoft-mpi
  - To configure project properties, you can use tutorials from: https://medium.com/geekculture/configuring-mpi-on-windows-10-and-executing-the-hello-world-program-in-visual-studio-code-2019-879776f6493f
- In case of difficulties with access to a working computer, you can access the torus server.