Exercise 4

Solution 4.1 (Hello MPI)**.** In this task the usage of basic MPI-primitives shall be exercised. Therefore you find an archive (Moodle **Exercise_04.zip**) with a directory **hello_MPI** and a to-be-completed code of a MPI ping-pong program.

a) Modify the program, such that the first process sends his console input to the second process (ping), which shall invert it ($x \to -x$) and send it back to the first (pong).

b) Modify the program such that each message contains a random number of elements.

Hint: You may have to announce the number of elements in a separate message prior to the actual message.

c) How does your program behave when it is launched with

i) one prozess,

ii) three processes, and

iii) with 16 processes?

Modify your program such that it works for all processe counts without error. All processes should participate in the message interchange similar to two processes.

d) Try solving part b) again, but this time without announcing the size of the main message explicitly.

e) Use your program to transmit message of varying sizes. What is the effect of the message size on the program runtime and how much time is spend inside MPI-routines?

Hint: Use the function MPI_Wtime() for time measurements.

Solution: A possible solution can be found in Moodle in archive **Solution_04.zip**.

Task 4.1 (The MPI Standard)**.** With this assignment you will learn about the MPI standard and how to use it. Please download yourself a copy of the MPI 3.0 report which represents the current MPI standard: `http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf`.
Hint: Browsing chapter 3 and Google may be of assistance.

a) What is an Message Envelope and where is it defined? What are the key elements? **Solution**:
The fields of an MPI message capable of distinguishing messages. It is defined in the MPI standard on page 27 in section 3.2.3. The fields are *source, destination, tag* and the *communicatior*

b) What communications modi are applicable to point-to-point messages? **Solution**:
Four communication modi are available for point-to-point messages: standard, buffered, synchronous and ready-mode. While the standard mode is always applicable and must be considered as blocking, the buffered send copies the message to an existing buffer.

Synchronous mode the send will only return once the matching receive has completed. In ready mode the matching receive must have been posted. All modi are indicators to the MPI library to enable optimizations of the communication.

c) How many methods (functions) are there to send a single message to a single receipient? What is the difference? **Solution**:
There are more than 10 different MPI routines that initiate the sending of data. The following routines that most common send routines: MPI_Send,MPI_Bsend,MPI_Ssend, MPI_Rsend, MPI_Isend, MPI_Ibsend, MPI_Issend, and MPI_Irsend.

d) What is an MPI request? **Solution**:
An MPI request is an opaque object to identify communication operations and match the operation that initiates the communication with the operation that terminates it.

e) Which MPI function are being used in conjunction with MPI_Mprobe? What is the use of those functions? **Solution**:
An MPI_Mprobe matches a specific message without receiving it. This "matched" message can no longer be received without the "match-object" and must be received using special routines such as MPI_Mrecv or MPI_Imrecv.

f) Is there a C++ binding for MPI 3.0? How do you use MPI in C++? **Solution**:
The C++ binding has been removed from the standard and does not exist as of MPI 3.0. C++ users can rely on the C-interface.