# Message-Passing Programming (2019-2020) Coursework: Code Submission

In addition to your report, you are expected to submit the code you have written for your Message-Passing Programming coursework. These guidelines explain what you should submit, and how it should be formatted. If you have any questions, please ask **Dr David Henty** (d.henty@epcc.ed.ac.uk) or **Dr Oliver Brown** (o.brown@epcc.ed.ac.uk).

## What should be submitted?

You are expected to submit a working copy of the code you have used to generate the results presented in your report. This is important: please do not make edits to your code after you have finalised and submitted your written report.

This code should be **one** of the following:

- A working message-passing parallel version of the percolation problem using a **two-dimensional domain decomposition**.
- Or, a working message-passing parallel version of the percolation problem using a **one-dimensional domain decomposition**. Note that you should submit this *only* if you are unable to develop a working version with a 2D domain decomposition. In this case:
  - You should explain in your report the issues you faced writing the 2D version.
  - You should also submit whatever code you have for the 2D implementation.
  - Any correctness and performance results in your report should come from the working 1D version, i.e. your Case Study solution.

Also included should be a **short markdown README** file, with the title **README.md**. In this file you should include the following:

- A **brief** description of the structure of your code:
  - Where are the source files located?
  - Where are the header files located?
  - What functionality is encapsulated in which files?
- How to build the code **on Cirrus**.
  - It **must** build with the mpt/2.18 and intel-compilers-18 modules loaded.
- Instructions on how to run the code on *P* processes, including any command line arguments. The first argument **must** be the random number seed as in the supplied serial code.
  - As submitted, your code **must** at a minimum be able to run using 16 processes on a grid size of 288 x 288 with the random number seed specified at run time.
  - By default, the code should run with $L$ = 288 x 288 and $\rho$ = 0.411. Specifically, if your program is called **percolate**, issuing the command **mpirun -n 16 ./percolate 1564** on the login node **must** run the code on 16 processes using a random number seed of 1564, a grid size of 288 x 288 and density $\rho$ = 0.411.

    - This may be adjustable with additional arguments (e.g. you could accept *L* and/or ρ as additional optional arguments), but **the defaults must be** $L = 288$ and $\rho = 0.411$.
    - If changing either the problem size or the number of processes requires modification of any source files, this **must** be clearly explained.
    - The code must produce an output file called "map.pgm" with a call to "percwrite", although you may wish to suppress file output for large simulations (e.g. $L > 1024$).
- Any restrictions on the running of the code.
    - For example: if the number of processes, *P*, must evenly divide the grid size, *L*.

Finally, note that this short README should **not** include any performance or correctness results, implementation details or code design – these should all be included in your report!

## Submission format.

We very strongly recommend that you use a version control system, such as git, while developing your code. Your remote repository **must** however be **private**. You can obtain free, private repositories from the University of Edinburgh Research Services GitLab (https://git.ecdf.ed.ac.uk/users/sign_in), GitHub through their Education pack (https://education.github.com/students), and BitBucket (https://bitbucket.org/product/).

The code **must** be uploaded to **Learn** as a single .zip or .tar archive. The archive **must** have the following naming convention: MPP1920-BXXXXXXX{-yyyy}.{zip|tar}

BXXXXXXX should be substituted by your exam number. {-yyyy} is an optional, hyphen-separated text string – for example, you may include the repository branch name here, which is the default if you download from GitHub or GitLab.

The archive file **must** unpack to a directory with the same name as the archive. For example, issuing **tar -xvf MPP1920-B1234567-master.tar** results in a directory called MPP1920-B1234567-master/ being created, **not** the source code appearing in the present directory.

The README.md file **must** be at the top level of your source code.

When you upload via Learn's **Turnitin** submission system, your "Submission title" **must** match the file name, e.g. **MPP1920-B1234567-master**

Finally, please ensure that your name is **not** included anywhere in the source code, archive or README file. Only your anonymous exam number should be visible to the marker.