# Deadline

26th November 2017

# Overview

The main aim of this study is to develop a **working** `Fortran` algorithm. It is an attempt for **mesh framework** for the numerical studies on fluid mechanics.

Potential modules:

- Mesh generation (diff. types)
- Reading/Writing meshes in different formats.
  - .vtk, pvtk, .nc, .hdf5
- Mesh analysis (mesh quality)

# Steps

- Choice a task for your module (derivative, integration, linear solver, statistics, etc)
- Develop your independent module, with it's routines
  - At this step your code should be compiled with the `make` command, means that you need to update `CMakeLists.txt` according to your need.
- Integrate your module to the others
  - **ATTENTION:** Every module has to use `module precision` and `module class_field` (at least)
- Develop your tests/demonstration program(s).
- Evaluate performance of the code you have written, feel free to choice aspects you like to evaluate (efficiency, accuracy, etc.)

# Details and Starter Codes

4 different parts of code is already provided to you.

## class_mesh

This is the base to hold derived type `mesh`. Your modules should use this type `mesh(s)` to store a mesh . And produce it's result(s) with `mesh(s)`. So anyone can modify this to improve their modularity. Please realize that; grids, velocity fields, derivatives, etc. should be implemented as `type field`.

## class_io

Input/output module, which reads/write with `NETCDF4` format. Already can read/write current type `mesh` which is incomplete leaving you the room to shape it to your needs.

### module precision

Very simple module to tune the precision of the whole code. Simple to implement. No need to change anything.

**ATTENTION:** All variables in any part of the code has to be defined with it.

### CMakeLists.txt

Please see the example available here. You can use parts of provided file directly. You still need to introduce your modules and executable(s) into `CMake`.

# Data

It is given as `tests/test_IO/statsi950.nc`. Also how to read and write such file is demonstrated with `test_IO_01_netcdf.f90` *(not completed yet)*.

# Report

**No special report expected.**

However, Doxgyen documentation, other inline comments, refactored variable names, README page, wiki Page(s) of this repo will be considered as a report, and they will be graded.

**Please provide examples of the output of your code simply using ">" pipe. So, I can be sure that, I run/compile your code in the way you have done.**

# Submission

**There is no special report submission required.**

All documents, and code base are placed here in this repository.

# Grading (tentative)

- 5 pts: Working code.
- 5 pts: Create a modular, working code.
- 5 pts: Report/Documentation/Comments with design decisions and evaluation.

  *This part includes, inline comments, doxygen comments, analyzes in comments, structure of your code, your choices for routines, functions, derived type etc.*
- 5 pts: Other contribution (README, wiki, CMakeList, Makefile, detailed documentation via Doxygen or other's module, etc.).

# Extra Credit (Up to 5 pts)

For all extra credit, be sure to document/analyze them clearly as well.

- Using external library (Lapack, PETSc, etc.)
- Parallel version of your code (MPI, OpenMP)
- Improvements on the project manner (Travis CI, Docker, etc.)
- Include the doxygen output as github page. https://pages.github.com/
    - See details: https://help.github.com/articles/user-organization-and-project-pages/

# Some Advice

*this section will be updated regularly, depending on the problems we diagnosed during the process*

- Choose a simple topic for your `module`. You can start to do something, if you feel that is getting complicated, just **simplify** it. This is not research contest. Only aim here is to learn how to use the tools efficiently and correctly in a simple scenario.
- Think about structure of your module before you start to write. Develop your strategy to achieve required modularity.
- Communicate each other, follow the other pieces of the code. Iterate the development process among each other. Commit your changes with your username to the github.