Scientific Computing for Mechanical Engineering [4EM30]
— *Instruction* —

Getting started with Git and GitHub

Proper management of your programming project poses a significant challenge, in particular when multiple people contribute to the project! To support software development control, so-called version management systems are used. In this course we will use the version management tool `Git` and the related online software development platform `GitHub`:

`https://github.com/`

`GitHub` has excellent online documentation. The tutorials at

`https://guides.github.com/`

will help you to get started with `GitHub` instantly:

1. Work through the `GitHub` "Hello World" tutorial on the guides page and create a public (and hence free!) `hello-world` repository on your own `GitHub` account. Explore the "Graphs" tab in `GitHub`, and in particular the "Network" sub-tab, and try to recognize the actions you performed in the tutorial.

Now that you have acquinted yourself with some of the basics of `Git` and `GitHub` it is time to interact with `GitHub` from the place where you will develop your code: your own computer! To get started, go to the official `Git` website[1]

`https://git-scm.com/`

and download and install `Git` by following the download link and selecting the appropriate operating system (most likely 64-bit Windows). Plenty of documentation is available through the `Git` site, which you will have to consult while working with `Git`. In particular the first two chapters of the online `Git` reference manual

`https://git-scm.com/doc`

are highly recommended to read, as is the chapter on `GitHub`.

In this instruction we will primarily use the `Git Gui`, which should now be installed as a program on your computer. We will use the `Git Gui` tool to make additional changes to your `hello-world` repository:

[1] Take a look at the bottom of the `Git` page to get an idea of the kind of companies and projects that use `Git` for their developments.

2. Open the `Git Gui` tool and select the "Clone Existing Repository" option. You can find the source location of your `hello-world` repository on your `GitHub` page (should be something like `https://github.com/<username>/hello-world.git`). Select an appropriate folder on your computer to copy the repositoty to (the target directory). Click the "Clone" button to copy the repository. Note that a folder with the contents of your repository has now been created. In addition, a `Git Gui` window with a status view of your repository is opened.

3. In the `Git Gui` window (which you can re-open when you restart the `Git Gui` application) select the "Visualize All Branch History" option in the "Repository" drop-down menu. Study this history carefully, and identify the operations that you conducted on the repository through `GitHub` (creation of the `readme-edits` branch, editing it, merging it back into master, etc.).

4. Go to the folder with your repository on your computer and open the "Readme.md" file in a text editor. Make some modifications to the file and then close it. Now go back to the `Git Gui` and select `Rescan` in the `Commit` drop-down menu (or press `F5`) to refresh your window. Note that the "README.md" file is now listed under "Unstaged Changes". Your modifications to the file are shown when the "README.md" file is selected. Now create a new text file named "new.txt" in your repository and verify that it also is listed in the "Unstaged Changes" section.

5. So far your changes are not "version managed". To do this, select the unstaged files and stage them for committing by selecting the `Stage To Commit` option in the `Commit` drop-down menu. Verify that both files are now listed under "Staged Changes". Now simply insert a "Commit Messagge" and press the `Commit` button to version manage your changes.

6. Inspect the history of your repository and verify that your local branch "master" is one commit ahead of your remote "master" branch on `GitHub`. Use the `push` button to synchronize the remote branch with your local branch, and inspect what changed in the repository history. Go to your online `GitHub` repository and verify that your changes are also available there.

You may have noticed that in the above workflow we directly worked on the master branch. When working on a project by yourself this can be very convenient. However, when working on a project with multiple people it is better to make your changes in a branch and merge these with the master branch by means of a Pull Request. This will avoid the occurence of so-called merge conflicts.

7. Use the Git Gui to clone the Instructions repository

   `https://github.com/TUE-4EM30/Instructions.git`

   to your computer. Note that this will give you the same files as obtained in the previous instruction by downloading the zip file. However, as we will see in the next instruction, you can now easily obtain the most recent version of this repository.

Exam scores

The scores of students that took part in an exam can be evaluated by a small program. The grades are listed in a test file that is read by the program, which then calculates the average grade, the standard deviation and prints a histogram. The code can be found in Pelles C project `example1`, which is located in the `Instruction2` folder of the Instructions repository that you just cloned from `GitHub`. While working on this assignment you can commit your code changes to your local repository using the `Git Gui`.

Unfortunately, the program has not yet been completed. It contains a number of syntax and semantic errors. In addition, some of the routines have not yet been implemented.

1. Before compiling and running the program, read the code carefully and see if you can identify the errors. A number of errors (less than 10) have intentionally been added to the code. List the errors in a table as shown below.

	Line number	Error	Compiler message
1 2 ...			

   Next, you may try to compile the code. Add the newly found errors in the table and mention the compiler messages (warnings and errors).

2. Debug the code and make sure that both the main code and the test code run properly (as far as it was completed).

3. Implement a function to calculate the standard deviation of the set of grades:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}\,. \tag{1}$$

   where N is the size of the sample, x_i the value of item i and $\bar{x}$ the average of the sample.

4. Implement a test for this function in the project `example1test`.

5. Complete the routine `fill_histogram` to calculate the histogram. The histogram has 12 intervals:

   1 : $0.0 \leq x < 0.5$
   2 : $0.5 \leq x < 1.5$
   .....
   11 : $9.5 \leq x < 9.9$
   12 : $x = 10.0$

   Implement a new function `print_histogram` to print the histogram to the screen in the following format:

```
0.0-0.5 : 12 #####
0.5-1.5 : 34 ###############
1.5-2.5 : 33 ###############
2.5-3.5 : 50 ##########################
....
etc.
```

where first the interval range is printed, then a semi-colon, followed by the number of items in the interval and a bar composed of pound keys to visualise the number of items in the interval. Scale the length of this bar in such a way that the longest bar is 40 characters long.

6. Design and implement a test for this function.