Exercises

General instructions

a) only submit code that compiles (in the case of C++) and runs

- otherwise you will not be able to complete the course

b) document your code appropriately

- I have to be able to understand why you write your code in the way that you do
- there are explicit questions, to which I expect responses (either as comments on the code or in separate text files)

c) in general, follow the directory structure that you will see used in the following

- exceptions: some code will be worked on in the context of multiple exercises
  - in these cases, a different naming convention will be used, and the way to separate different iterations will be different; details will follow

Exercise 0.1

The aim of this exercise is to get you started on a few topics you will need further on: git/gitlab and compilation.

git is an open-source, fast, scaleable distributed revision control system that is heavily used in large projects. **gitlab** is a service that can be used to provide access to git repositories and also "hosts" your code. We will use these for handing in exercises, using in particular the C&CZ gitlab server (see wiki page):

a) start by creating an account on this server by logging on to it
   - once you have the account, let me know so that I can give you access to the repository created for this course
     - I added those of you that I (think I) already found on the server

b) follow instructions for creating an ssh key

Exercise 0.1 (continued)

c) create a new (empty) repository on gitlab for you to store your completed code in, again using the instructions on the wiki page

- do NOT add files or make any commit at this point (contrary to the wiki page's instructions)
- and provide me (username: filthaut) with guest access on gitlab

d) retrieve my initial repository

- git remote add upstream https://gitlab.science.ru.nl/filthaut/cds-advanced-programming.git ← makes it referrable to as "upstream"
- git fetch upstream ← useful also later to synchronise repositories
- git checkout -b master -- track upstream/master

  ← for initial checkout; for later modifications (from my side) you will want to use "git merge"
- git config branch.master.remote origin ← make your gitlab repository the target for updates

Exercise 0.1 (continued)

e) At this point you should have a working directory with an **ex0.1** subdirectory. cd to it and follow the instructions in README.md (also visible here)

- create a text file (let's call it results.txt) detailing your findings
- git add results.txt
- git commit $\longleftarrow$ add changes to your local repository
- git push -u $\longleftarrow$ and synchronise your gitlab repository with it

Notes:

- my repository is not finished at this point, so "git fetch upstream" and "git merge" will be useful later
- more documentation on git can also be found from the C&CZ wiki page
  - also (for git): "man git"; "git help <topic>" (where topic = add, commit, fetch, pull, push, etc.)

Exercise 0.2

The aim of this exercise is to show that Python is already doing more than “just” interpreting the user level Python code. Notably, packages like numpy and scipy exploit the power of compiled C code behind the scenes, and this is especially helpful in the case of large arrays. It is relatively easy to test this last feature, by performing the same task with different configurations.

a) Use numpy to generate a large number of random numbers, in one shot.
   - I have done this with up to 10^9 RN — but note that this requires a substantial amount of memory (4 GB), and you may want to verify how much memory you can afford for your task

b) Do the same but in some large number of times.

Add both the Python macro(s) and the timing output comparison to the repository, in a new subdirectory ex0.2

- “git add ex0.2”; “git commit”; “git push -u”