TP 0: shell + hello world

1 Discovering the shell

The goal of this exercise is to get familiar with the so-called "shell" running in a command window. All of the steps below except for the first one could/should be done without using the mouse (if you wish to fully apply to this rule it implies not using a browser to search for help on the internet...). Help within the shell can be obtained using the `man` command (a shorthand for *manual*), and the shell itself is called *bash* (shorthand for *Bourne again shell*), so `man bash` will give you access to the (huge) description of it. Help on individual commands can be obtained by `man` followed by the name of the command.

1) Open a terminal window.

2) Read the manual of the commands `cd` and `pwd`.

3) Navigate to the root directory `/`, read the manual of the command `ls`, and then list the content of the root directory (all entries and with long listing format). Are these listed entries files or directories ? Who is the owner of these and would you be able to read and/or write over them ?

4) Navigate back to you home directory, list all of its entries using `ls`, and place them into a file called `file1.txt`. For that purpose, explore the possibilities of redirecting the shell output(s) into files (inside the bash manual search for the section called REDIRECTION, searches like in the Vim editor are done using the slash key `/` followed by the searched string).

5) Read the manual for the command `mkdir`. From your home directory, create a subdirectory named `MA016`, and inside the latter another one named `tp0` (can you do both at once ?).

6) Read the manual for the commands `cp` and `mv`. Make a copy of the file `file1.txt` inside `MA016/tp0`, which you will name `file2.txt`. Then move `file1.txt` into `MA016/tp0`. Navigate into `MA016/tp0` and check that both files inside it are identical (using the `diff` command).

7) Edit `file2.txt` using an editor (`vim` and `emacs` are top choices), apply some changes, and then observe how the difference is reported by `diff`.

8) Erase both of these files now, so that we have a clean directory for real work.

Other frequently used shell commands include : `pwd`, `cat`, `more`, `ps -aux`, `top`, `grep` or `egrep`. The use of pipes `cmd1 | cmd2` that bridge the output of `cmd1` to the input of `cmd2` is also a powerful feature to be aware of.

You will find tons of information online about the shell, feel free to now use a browser and a mouse in your learning process (but remember about the existence of the `man` command, and the fact not all online resources are exact...).

2 Hello world in C

1) Create the source code file `hello.c` for a program that will simply print "Hello, World!" to the screen.

2) Compile that program into an executable named `hello`, using the GNU C compiler `gcc` directly.

3) Compile it instead using the CMake tool (documentation online). This may seem tedious at first, but it will save you a lot if time when your projects will get bigger, with many source files and/or third party code.

This is just the starting point of your journey with C/C++. There are also tons of information online, and some good printed books to (ask me depending on your present knowledge of these langages). For a quick reference, `cppreference.com` and `cplusplus.com` are recommended online sources.

3 Loops and formatted output

Write a C program that takes as command line argument an integer n and then prints the first n lines of Pascal's triangle in the terminal:

$$\begin{array}{llll} A_{0,0} & & & \\ A_{1,0} & A_{1,1} & & \\ A_{2,0} & A_{2,1} & A_{2,2} & \\ \vdots & \vdots & \vdots & \ddots \end{array}$$

using the recurrence formula $A_{i,j} = A_{i-1,j-1} + A_{i-1,j}$ (if $0 < j < i$, and 1 otherwise).

4 Reading and writing into a file using C

1) Build a C program, with executable called `read_file`, which reads the content of a text file (whose name is a command line argument) and print it line by line in the terminal. Test it over the file `test-tp0.txt` included within this TP0 directory.

2) Let n be a positive integer, $x_{\min} = -6\pi$, $x_{\max} = +6\pi$, and $\Delta x = (x_{\max} - x_{\min})/(n-1)$. For $j = 0, \ldots, n-1$, set $x_j = x_{\min} + j\Delta x$ and $y_j = \sin(x_j)/x_j$. Write a program that takes n as input and writes a text file formatted as follows:

$$\begin{array}{ll} x_0 & y_0 \\ x_1 & y_1 \\ \vdots & \vdots \\ x_{n-1} & y_{n-1} \end{array}$$

with a tab separating elements in the same line, and linebreaks to separate lines.

3) Make a graphical representation of the function $x \mapsto \sin(x)/x$ on the interval $x \in [-6\pi, +6\pi]$ using GNU Gnuplot tracing program (see `http://www.gnuplot.info`).

5 Using Git

I have set up a Git repository for you to save your files during the whole semester. That not only provides a backup, it will teach you how to deal with version/history in larger projects, and also collaborative work.
The repository is hosted on Github, and you would traditionally clone it using :

```
git clone https://github.com/didiersmets/MU4MA016_Students.git
```

That would only give you read access to it. I have therefore created a so-called token giving you (limited) write access until the end of January 2025. For that purpose, from your home directory in a terminal window clone the repository using the command :

```
git clone https://[token]@github.com/didiersmets/MU4MA016_Students.git
```

where you replace `[token]` by

```
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
```

That will create a directory called `MU4MA016_Students` inside your home directory, and you can check that the token was saved (with other info) inside the `.git/config` file in that directory. Inside that same directory, within Students, create a subdirectory whose name is your family name (better avoid blank space in the name), in the sequel I call it NAME. Organize your source files to your liking into that NAME directory, and then upload it to Github. For that purpose you will need to :

1. Set your name and email into your local git config : from inside the `MU4MA016_Students` directory launch the commands

   ```
   git config --local user.name "yournamewithoutspace"
   ```

   and then

   ```
   git config --local user.email "youremail"
   ```

   You can check that this is recorded in the `.git/config` file and won't be needed anymore for future uploads.

2. Next, add your directory for the next commit :

   ```
   git add NAME
   ```

   You can check which files were indeed added by

   ```
   git status
   ```

   At this point, these files are only in *staged* state, and the changes could be revertible.

3. Commit your changes with a commit message :

   ```
   git commit -m "The first commit of [yournamehere]"
   ```

   The modification to your local repository is now active.

4. To sync it with the remote repository (i.e. on Github), finally push your commit there :

   ```
   git push
   ```

   By visiting `https://github.com/didiersmets/MU4MA016_Students` you can verify that your commit appears (along the ones of your class mates, for those who did it before you).

Take the time to lear a little more of Git, either through the `man` command or using online tutorials. The goal is to get familiar with that fantastic tool. Note that `git pull` will bring back the last centralized version of the repository, including your class mate files ! (on purpose, you can then learn by glancing at your class mate's code too).
You are requested to push to that NAME directory regularly (e.g. at least once a week) for uploading your work, for this and the later exercise sessions. That is a safe practice, and also a way for me to supervise your work/progress.