C++ and Linux tutorial

1. A Linux terminal is a text interface that allows you to write and execute commands. The goal of this exercise is to learn a few simple commands.

   a) Open a new terminal (shell)

   b) Create a new directory named *test*: `mkdir test`

   c) Go into the directory: `cd test`

   d) Start a text editor software to write a program in C++ (emacs, vim or gedit): `vim test.txt`

   e) Type something, save the file and exit the editor

   f) A few other useful Linux commands are:

Command	**Utility**
`pwd`	Show path of current directory
`mkdir` *name*	Create directory *name*
`cd` *name*	Go into directory *name*
`gedit`	Start the text editor
`ls`	List all files and directories in current directory
`ls -l`	Include additional information in the listing
`cp`, `mv`	Copy/move files
`man` *command*	Open the manual of a *command*
command `-h`	Get help on the syntax of a *command*
`g ++`	Compile C and C++ programs
`exit`	Exit the terminal
`tar -xvzf` *name*.tar.gz	Uncompress file with extension tar.gz
`tar -cvzf` *name*.tar.gz	Create zipped file with extension tar.gz

2. **Introduction to C++** | The goal of this task is to write a first C++ program.

   a) Open a terminal

   b) Create a new directory called *hello*

   c) Go into the directory

   d) Use a text editor to open a C++ file called `hello.cpp`

   e) Type the following code:

```
#include <iostream>
using namespace std;
int main(){
   cout<<"Hello World"<<endl;
   return 0;
}
```

f) Compile the code: `g++ hello.cpp -o hello.x`

g) Run the code: `./hello.x`

h) Change the output text, compile and run the program again

3. **Another C++ example** | The goal is to write a code that generates a table with the values given by a parabola.

   a) Open a file parabola.cpp and write the following code:

```
#include <iostream>
using namespace std;
int main(){
   for(int i = 1; i<=10; i++){
      double y = i*i; // Create new variable
      cout<<i<<"\t"<<y<<endl;
   }
   return 0;
}
```

   b) Run the program saving the output to a file *parabola.dat*: `./parabola.x > parabola.dat`

4. **Simple arrays** | Implement a program that defines an array with the following values

   $\{10.5, 9.3, 11.4, 10.9, 13, 8.4, 9.2, 8.9, 10.3, 11.2, 12.1, 8.4, 9.2, 9.9, 10.1\}$

   The program should run over all values and print them to the screen. Then it should ask the user to enter a number between 1 and 15 and print the corresponding number of the array.

5. **Calculate mean values and standard deviation** | Change the program you wrote on the previous exercise to calculate the following quantities:

   a) Mean value of the numbers in the array

$$< x >= \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1}$$

   b) Standard deviation

$$\sigma = \sqrt{var}, \qquad var = \frac{1}{N}\sum_{i=1}^{N}(x_i - < x >)^2 \tag{2}$$

6. **Conditional statements** | Using the same program as in the previous exercises, define the following array (it should have the same size as the previously defined array)

   $\{1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1\}$

   Loop over the entries of the array and whenever you find an entry with the value 1 print the corresponding entry of the initial array. Then for all entries marked with 0 (or 1) calculate the mean value and the standard deviation.

7. **User defined functions** | Let's repeat exercise 5 a) but this time creating a function that takes as input the array and its size and returns its mean. Some notes on functions below.

   The syntax to defining a function in C++ is:

```
returnType functionName (parameter1, parameter2, ...){
//function body
}
```

   where `returnType` is the type of the variable that will be returned and `parameter1, parameter2, ...` are the arguments of the functions (no arguments is also possible). In this case `parameter1` should be the array and `parameter2` should be the size of the array. When giving arguments as input to a function their type must also be specified.

   Function should be defined before the `main` function and then can be called from inside the `main` function.

8. **Dictionaries in C++ ?**

   The C++ direct equivalent of dictionaries in python are maps. Each entry corresponds to a pair *key, value*. For this exercise let's define a map containing grades normally distributed with mean 15 and standard deviation 5. In this case the keys are the names of the students (or the student number for example) and the values are their grades.

   a) Print all entries in the map. By default they should be ordered in alphabetical order of the keys.

   b) Order the map from highest to lowest grade and print the entries again. (Tip: add the dictionary entries to a vector of pairs and then use the *sort* function)

9. **Plotting with gnuplot** | Let's go back to exercise 3. The code we wrote produces an output file called `parabola.dat`. It contais a list of x and y values that define a parabola. We will use this file as input to gnuplot.

   a) From the terminal open gnuplot by typing `gnuplot`. This will start a gnuplot session. Then plot the points from the `parabola.dat` file by typing `plot "parabola.dat"`

b) A more convenient way of using gnuplot is to write a small script with the commands that we want to execute and then run it using: `gnuplot Ex9_gnuplot_demo.gnu`. This script contains the base code to read the `parabola.dat` data file and draw a plot. Change the script such that the plot has a title and axes labels.

c) Fit the data points with a linear function. Display the fitted function and the data points in the same plot.

On Linux/Mac environment

1. Open a terminal prompt and check if you have a `g++` compiler installed: type `dpkg --list | grep compiler` and check if `g++` shows up on the list. If yes, you are all set and can continue with the tutorial.

On Windows

1. Download and install MobaXterm which is a terminal prompt emulator for windows: https://mobaxterm.mobatek.net/

2. Start the MobaXterm application and start a new local session. You should now have something similar to a Linux prompt (black background with letters)

3. Install `g++` compiler: in the command line type `apt-get install gcc-g++`

4. Install a text editor (I'll be using emacs but many other are available so feel free to use your favourite one): type `apt-get install emacs-w32`