Laboratory 1: Introduction to C++

This is a typical script that you will be working with at each laboratory session. To work with such a script efficiently, follow the guidelines below:

1. The script is not a step-by-step tutorial. It introduces the problem but it is your task to solve it.
2. If you get stuck, make sure that you read the *Help* section at the end of the document.
3. If you still have problems, ask the lecturer, demonstrator or fellow students for help.
4. Try to do as much work as possible in the class, where it is easier to get help.
5. If you need help when working at home, use the Blackboard discussion board. I tend to reply to my emails, but sometimes they get buried, so don't worry if I don't reply to your email and send it again. My office is Room 324, and Shatah's is 320.
6. Finish all assignments at least a week before the deadline. If you get stuck, you will still have one week to ask for help.

Task 1: Using CMake

It can be extremely complicated to use the An integrated development environment (IDE) to maintain the project files, etc. particularly if most students use Windows, some use Mac, and others use GNU/Linux. There is a ZIP file on Blackboard with 4 (almost) empty C++ source files. There is also a 'CMakeLists.txt' file. It is a good practice to organise your source code using directories, etc. Do not compile code in the same directory as your source code. It can get messy... I recommend the following organisation:

```
ICP3038
   |-Lab-01
   |     |-bin (where you compile the code)
   |     |-doc (where the documentation is)
   |     |     |- *.tex
   |     |     \- *.pdf
   |     \-src (where the source code is)
   |           |- *.h
   |           |- *.cxx
   |           \- CMakeLists.txt
   |-Lab-02
   |     |-bin (where you compile the code)
   |     |-doc (where the documentation is)
   |     \-src (where the source code is)
```

DO NOT COMPILE YOUR CODE RIECTLY IN THE `src` DIRECTORY. Extract the ZIP file, and set up the compilation environment using `cmake`. In the GUI, the source directory corresponds to the direction in which 'CMakeLists.txt' is. The binary directory is where the code will be compiled. Often, we call this directory 'bin'. Once the paths are set, press 'Configure'. The first time you run the configuration tool, you have to select a generator. If you want to use MSVC++ in the lab, make sure to use *Visual Studio 15 2017 Win64*. For Mac OS X, you may want to use the Xcode generator. For Linux, you may choose Makefile. Once the configuration step is over, press 'Generate'. The project files are now ready in the 'bin' directory. You can compile the code using your preferred IDE.

Task 2: cout/cin

Your task is to program a command-line application that:

1. Reads two integers (`i` and `j`) from the standard input (i.e. the terminal) and outputs their product into the standard output (i.e. the terminal).
2. Reads two strings (`s1` and `s2`) from the standard input and outputs them onto the same line into the standard output.

The C++ objects and methods that you need to use are:

```
ostream& operator<< (val)
istream operator>> (&val)
std::cin
std::cout
string
```

`cout`, `cin`, `<<`, `>>`, `are defined in \verb¡iostream¿+ header file. The 'c' in` `cout` refers to "character" and 'out' means "output": `cout` means "character output". It is used along with the insertion operator (`cout¡¡+`) to display a stream of characters. The general syntax is:

```
cout << varName;
```

Or

```
cout << "Some String";
```

The extraction operator can be used more than once with a combination of variables, strings and manipulators (like endl):

```
cout << var1 << "Some String" << var2 << endl;
```

Note that `endl` means “end of line”. It replaces `\n` in the printf function of C.

`cin` means “character input”. It is used along with the extraction operator (`cout¿¿+`) to receive a stream of characters. The general syntax is:

```
cin >> varName;
```

The extraction operator can be used more than once to accept multiple inputs as:

```
cin >> var1 >> var2 >> ... >> varN;
```

`string`is defined in `<string>` header file. It is a class that handles strings of characters.

Task 2: Size of basic types

Your task is to program a command-line application that produces the output as follows:

```
The size of char is:        1  bytes
The size of short is:       2  bytes
The size of int is:         4  bytes
The size of long is:        8  bytes
The size of long long is:   8  bytes
The size of float is:       4  bytes
The size of double is:      8  bytes
The size of long double is: 16 bytes
The size of bool is:        1  bytes
```

To get the size of a `char`, just use `sizeof(char)`.

Task 3: Command line arguments

Your task is to program a command-line application that gets two integers from the command line arguments and swap their order to produce the output below:

```
Input 1st number: 25
Input 2nd number: 39
After swapping the 1st number is: 39
After swapping the 2nd number is: 25
```

You need first to read the two integers, display the first two lines, swap the integers using the function provided, and display the last two lines.

Note that in `int main(int argc, char** argv)`, `argc` is the total number of command line arguments passed to your program, `argv` is an array of C strings containing the arguments. `argv[0]` corresponds to the executable file, `\argv[1]` is its first argument, `\argv[argc-1]` is the last one.

To convert a C string into an integer, you need the `int atoi (const char * str)` function. It is located in `<cstdlib>` header file.

The C++ objects and methods that you need to use are:

- `cerr` for the standard output of errors (that's where error messages should be printed); and
- `double strtod (const char* str, char** endptr)` to convert a C-string into float. To learn about a standard function, it is always useful to go on `http://www.cpluplus.com/`. In the menu on the left-hand side, there is *Reference*. That's where you have to go. Hint, `strtod` is defined in the `<cstdlib>` header file. As an alternative, you could use the search button to directly go to the page about `strtod`. The advantage of the search method is that you don't need to know in which header the function is.

Task 4: Bubble sort

Your task is to program a command-line application that:

1. Reads a single integer N from the standard input, where N is the number of floating point values to be sorted.
2. Reads exactly N floating point values from the standard input. The numbers are separated by a new-line character.
3. Store the N floating point values into an array/
4. Sorts those values in increasing order using your own implementation of the bubble sort algorithm, swapping the values using the function *swap* written by you.
5. Prints out to the standard output the number of elements and then, in separate lines, all the numbers in the sorted order.

Example: Given the input file:

```
3
1.5
-2
0
```

your program should generate:

```
3
-2
0
1.5
```

Help

C++ resources

There are plenty of tutorials and resources for Java programmers who wish to learn C and C++. You are strongly recommended to research this topic on your own to gain better understanding of the differences between C++ and Java. This module is not meant to teach C++ but expects that you gain basic C++ skills as you solve the tasks at the laboratory classes.

Some on-line resources for Java programmers moving to C++:

1. Moving from Java to C++ — `http://www.horstmann.com/ccj2/ccjapp3.html`
2. C and C++ for Java programmers — `http://goo.gl/SxHYS`
3. C++ tutorials and documentation — `http://www.cplusplus.com/`

Installing CMake on your computer

It is strongly recommended that you install CMake on your PC so that you can complete your labs and assignments at home. You can find it at `https://cmake.org/download/`. Go to *Binary distributions* and choose your platform.

Installing an IDE on your computer

For the same reasons, it is strongly recommended that you install Visual Studio on your PC, Xcode on Mac, or Eclipse CDT on linux. Note that the installation of the development tools is often a difficult and time-consuming process. Do not assume that you can install the software a few days before deadlines. Install the software as soon as possible and let us know if there are any problems.

Windows: Installing Visual Studio Community

Download and install *MS Visual Studio Community* from

`https://visualstudio.microsoft.com/vs/community/`

Mac OS X: Installing Xcode

This setup will require installing Xcode, which can take up to 2-3GB of your hard drive. Make sure that you have sufficient hard drive space before your proceed.

Install *Xcode* from App Store. *Xcode* is a free application for Mountain Lion and later versions of OSX.

Linux: Eclipse CDT

Download and Install *Eclipse CDT* from `https://eclipse.org/cdt/downloads.php`. You will need a Java virtual machine as it uses Java in the background.