Part A - Introduction

Compiling Modular Source Code

Workshop 1 (out of 10 marks - 3% of your final grade)

In your first workshop, you are to sub-divide a program into three modules and compile the modules separately, on different platforms.

SUBMISSION POLICY

The “in-lab” section is to be completed **during your assigned lab section**. It is to be completed and submitted by the end of your workshop period. If you do not attend the workshop, you can submit the “in-lab” section along with your “at-home” section (a 30% late deduction will apply). The “at-home” portion of the lab is **due on Sunday January 24, 2016, before 11:59:59pm.**

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible to regularly backup your work.

LEARNING OUTCOMES

Upon successful completion of this workshop, you will have demonstrated the abilities

- to organize source code into modules, with header and implementation files
- to compile and run modular programs on different platforms
- to accurately describe the work you have done

ORIGINAL SOURCE CODE (THE SENEGRAPH PROGRAM)

SeneGraph is a program that receives several statistical sample values and compares those values using a horizontal Bar Chart.

Download the original source code of SeneGraph (Workshop1) from GitHub in one of the following two methods: (command line method is preferred)

1- Downloading the zip file and extract files:
   a. Open the following URL in a browser:
   b. https://github.com/Seneca-244200/BTP-Workshop1
      Click the "Download ZIP" button
   c. Extract the files in to your working directory
2- Cloning the repository using command line:
   a. If you are using your own computer, download the latest version of git from https://git-scm.com/downloads and install it using its default options. (this is done only once throughout the semester for all workshops)
   b. Issue the following command at command prompt in your working directory:
      `git clone https://github.com/Seneca-244200/BTP-Workshop1 <ENTER>`
      The URL for all the workshops are the same throughout the semester. The only thing that changes, is the workshop number. You can also find this URL on workshop page on GitHub.

IN-LAB SECTION (80%)

STEP 1:

1- Windows, Visual Studio (VS): compile and run and test the program:
   A. Open **`Workshop1/in_lab`** directory (that you just downloaded/cloned) and double-click on **`in_lab.vcxproj`**. This will open a Visual Studio (VS) project that is already created in the same directory. *If an option is given to you to for the version of Visual Studio (VS), select Visual Studio 13.*
   B. In VS open Solution Explorer *(click on View/Solution Explorer)* and then add **`w1_in_lab.cpp`** file to the project.
      You can do this by following this:
      - Right click on "Source Files"
      - Select "Add/Existing Item"
      - Select "**`w1_in_lab.cpp`**" from file browser
      - Click on "OK"
   c. Compile and run the program by Selecting "Debug/Start Without Debugging" or by pressing "Ctrl-F5".
2- Linux, Matrix:
   a. Transfer **`w1_in_lab.cpp`** to your matrix account (ideally to a designated directory for your btp200 workshops).
      Issue the following command to compile the source file, creating an executable called "**`w1`**":
      `clang++ w1_in_lab.cpp -Wall -std=c++0x -o w1 <ENTER>`

`-Wall:` display all warnings
`-std=c++0x:` compile using C++11 standards
`-o w1:` name the executable "w1"

b. Type the following to run and test the execution:
`w1 <ENTER>`

STEP 2:

Windows, Visual Studio (VS):

In solution explorer, add three modules to the project: seneGraph, graph and tools. seneGraph has only one cpp file. graph and tools have both cpp and header files:

- Add **`seneGraph.cpp`**, **`graph.cpp`**, and **`tools.cpp`** to "Source Files"
  (Right click on "Source Files" and select "add/new Item" and add a C++ file)
- Add **`graph.h`** and **`tools.h`** to "Header Files"
  (Right click on "Header Files" and select "add/new Item" and add a header file)

Divide the functions in **`w1_in_lab.cpp`** and copy them into the modules as follows:

Tools module will contain the **`menu`** and **`getInt`** functions. Copy the implementation of the functions to the cpp file and the prototypes to the header file. Make sure you include "**`tools.h`**" in "**`tools.cpp`**".
To test that you have done this correctly, you can compile the module separately; right click on **`tools.cpp`** and select compile from the menu. If the compilation is successful, most likely it is done right.

Note for compiling on Matrix:
The equivalent of this on matrix is to add "-c" to the compile command:
`clang++ tools.cpp -Wall -std=c++0x -c <ENTER>`
This will only compile tools.cpp and will not create an executable.

Graph module will contain the **`getSamples`**, **`findMax`**, **`printBar`**, and **`printGraph`** functions. Copy the implementation of the functions to the cpp file and the prototypes to the header file. Also add the define statement for **`MAX_NO_OF_SAMPLES`** to "**`graph.h`**". Make sure you include "**`tools.h`**" and "**`graph.h`**" in "**`graph.cpp`**".
To test that you have done this correctly, you can compile the module separately; right click on graph.cpp and select compile from the menu. If the compilation is successful, most likely it is done right.

Note for compiling on Matrix:
The equivalent of this on matrix is to add "-c" to compile command:
`clang ++ graph.cpp -Wall -std=c++0x -c <ENTER>`
This will only compile graph.cpp and will not create an executable.

`SeneGraph.cpp` file contains the `main` function. Include "`tools.h`" and "`graph.h`" in `seneGraph.cpp`.

All cpp files must include `iostream` and contain "`using namespace std;`" at the very beginning of the file.

Now remove `w1_in_lab.cpp` from the project. You can do this by right clicking on the filename in solution explorer and selecting "Remove" in the menu (make sure you do not delete this file and only remove it).

Compile and run the project (as you did before) and make sure everything works.

Linus, Matrix:

Create the five files (or upload them if you used VS) to your matrix account and issue this command to compile the code.

```
clang++ tools.cpp graph.cpp seneGraph.cpp -Wall -std=c++11 -o w1 <ENTER>
```

Run the program and make sure everything works properly.

Sample execution: (Red values are entered by user)

```
Welcome to SeneGraph
No Of Samples: 0
1- Number of Samples
2- Sample Entry
3- Draw Graph
0- Exit
> 1
Enter number of samples on hand: 3
No Of Samples: 3
1- Number of Samples
2- Sample Entry
3- Draw Graph
0- Exit
> 2
Please enter the sample values:
1/3: 30
2/3: 60
3/3: 100
No Of Samples: 3
1- Number of Samples
2- Sample Entry
3- Draw Graph
0- Exit
> 3
Graph:
********************* 30
****************************************** 60
```

```
********************************************************************** 100
No Of Samples: 3
1- Number of Samples
2- Sample Entry
3- Draw Graph
0- Exit
> 0
Thanks for using SeneGraph
```

IN-LAB SUBMISSION (80%)

On matrix run the following command:

`whoami <ENTER>`

and make sure the output of the command is your own user id; if not, report it to your professor and do not submit your workshop.

Then from the location of your files on matrix issue the following command and follow the instructions. Use the above sample execution values for your submission.

`~olivier.st-cyr/submit w1_in_lab <ENTER>`

AT-HOME SUBMISSION (20%)

A) Answer the following question:

Why do you think a modular approach to programming is preferred in industry compared to having all the code in one file? Explain your answer with an example.

Create a file called **`w1_at_home.txt`** and write your answer in it. Don't forget to save the file.

B) Do ***one*** of the following two:

1- If you have never created a console application using and IDE like Visual Studio:

> Go to following webpage:
> https://github.com/Seneca-244200/OOP-Videos
> and watch this instructional video:
> Creating simple Console App using Visual Studio
>
> In your own words, write in your **`w1_at_home.txt`** file the steps to create a simple console application in visual studio. Don't forget to save the file.

2- If you already know how to create a simple console application in an IDE:

> In your own words, write in your **`w1_at_home.txt`** file the steps to create a simple console application in the IDE of your choosing. Don't forget to save the file.

On matrix run the following command:

`whoami <ENTER>`

and make sure the output of the command is your own user id; if not, report it to your professor and do not submit your workshop.

Then upload the file to matrix and from the location of your file issue the following command and follow the instructions.

```
~olivier.st-cyr/submit w1_in_lab <ENTER>
```

It is strongly recommended to watch all three videos posted at https://github.com/Seneca-244200/OOP-Videos