Part A - Introduction

Compiling Modules

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

In your first workshop, you are to sub-divide a program into modules, compile each module separately and construct an executable from the results of each compilation.

LEARNING OUTCOMES

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

- to organize source code into modules, using header and implementation files
- to compile and link modular programs
- to distinguish the contents of a header and an implementation file
- to describe to your instructor what you have learned in completing this workshop

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 the workshop. 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 be assessed). The “at-home” portion of the lab is **due the day before you next scheduled workshop.**

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

You are responsible to regularly back up your work.

ORIGINAL SOURCE CODE (THE SENEGRAPH PROGRAM)

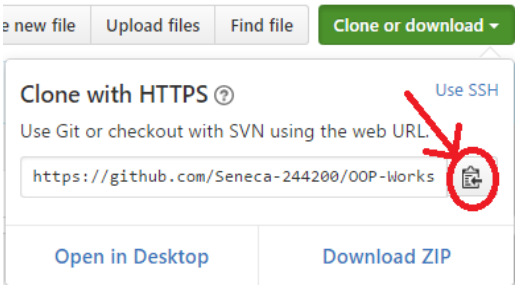
SeneGraph is a program that receives several statistical sample values and compares those values using a horizontal Bar Chart.

> *For this part, it is better to do this lab at school and on a lab computer since it has “Git” and “Tortoise Git” installed. If you do have “Git” or “Tortoise Git” installed on your own computer you can do this on your own personal computers. (“Tortoise Git” Installation guidelines are in the “at home” section of this lab”)*

1- Open https://github.com/Seneca-244200/OOP-Workshop1 and click on “Clone or download” Button”; this will open “Clone with HTTPS” window.

If the windows is titled “Clone with SSH” then click on “Use HTTPS”:

2- Copy the https URL by clicking on the button on the right side of the URL:

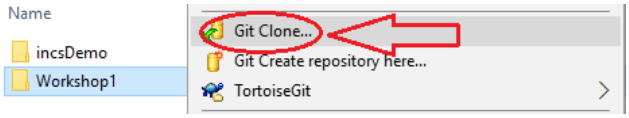


3- Open the File Explorer on your computer and select or create a directory for your workshops.

Now Clone (download) the original source code of SeneGraph (Workshop1) from GitHub in one of the following three ways: (methods 1 and 2 are preferred)

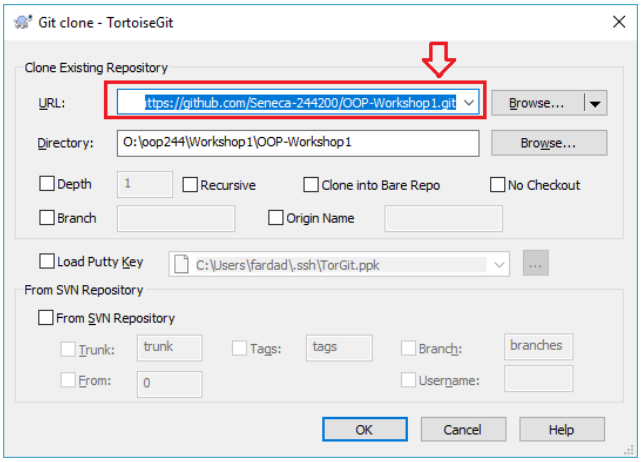
1- Cloning the workshop repository using Tortoise Git:

a. Right click on the selected directory and select “Git Clone”:



This will open the “Git Clone” Window with the URL already pasted in the “URL” text box; if not, paste the URL manually:

b. Click on OK.



This will create a clone (identical directory structure) with the one on Github on your computer; now you can open the directory OOP-Workshop1 and start doing your workshop. Note that this process will be repeated for all workshops and milestones of your project in this subject.

*If your professor makes any changes to the workshop, you can right click on the cloned repository directory and select **TortoiseGit / pull** to update and sync your local workshop to the one on Github without any need to download it again. Note that this will only apply the changes made and will not affect any work you have done in your workshop.*

2- Cloning the repository using command line:

c. Open the Git command line on your computer.
d. Change the directory to your workshops directory.
e. Issue the following command at the command prompt in your workshops directory:

```
git clone https://github.com/Seneca-244200/OOP-Workshop1.git <ENTER>
```

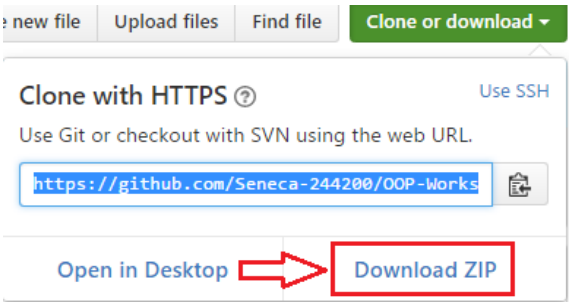
The URL for all the workshops are the same throughout the semester. The only thing that changes, is the workshop number.

This will create a clone (identical directory structure and content) with the one on Github on your computer; now you can open the directory **OOP-Workshop1** and start doing your workshop. Note that this process will be repeated for all workshops and milestones of your project in this subject.

*If your professor makes any changes to the workshop, you can issue the command: **`git pull <ENTER>`** in the cloned repository directory to update and sync your local workshop to the one on Github without any need to download it again. Note that this will only apply the changes made and will not affect any work you have done in your workshop.*

3- Downloading the repository using “Download ZIP” option:

a. Open https://github.com/Seneca-244200/OOP-Workshop1 and click on “Clone or download” Button” and click on “Download ZIP”



b. A zipped file copy of the workshop repository in Github will be downloaded to your computer. You can extract this zip file to where you want to do your workshop.

Note that, if your professor makes any changes to the workshop, to get them you have to separately download another copy of the workshop and manually apply the changes to your working directory to make sure nothing of your work is overwritten by mistake.

IN-LAB SECTION (50%)

STEP 1:

1- Windows, Visual Studio (VS); compile and run and test the program:
   A. Open **`OOP-Workshop1/in_lab`** directory (that you just cloned/downloaded) and click on **`in_lab.vcxproj.`** This will open a Visual Studio (VS) project that is already created in the same directory.
   B. In VS, (if not open already) 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. Upload w1_in_lab.cpp to your matrix account. (Ideally to a designated directory for your oop244 workshops).
   Issue the following command to compile the source file, creating an executable called "w1":
   `g++ 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 new modules to the project; seneGraph, graph and tools. seneGraph module has only one cpp file. graph and tools modules 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, 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:*
> ***g++ 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:*
> ***g++ graph.cpp -Wall -std=c++0x -c <ENTER>.***
> *This will only compile graph.cpp and will not create an executable.*

SeneGraph module 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.

Linux-Matrix:

Upload the five file to your matrix account and issue this command to compile the code.

```
g++ tools.cpp graph.cpp seneGraph.cpp -Wall -std=c++0x -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 (50%)

To test and demonstrate execution of your program use the same data as the output example above.

If not on matrix already, upload **tools.cpp, tools.h, graph.cpp, graph.h** and **seneGraph.cpp** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

```
~profname.proflastname/submit 244_w1_lab <ENTER>
```

and follow the instructions.

AT-HOME (50%)

PREPARING YOUR COMPUTER AT HOME FOR THIS SUBJECT

If you are working on your own Windows computer at home:

> Download Visual Studio Community from here:
>
> https://www.visualstudio.com/downloads/
>
> and Install it.
>
> When installing, make sure you select the custom installation and select Visual C++ in programming languages. Here is the “how to” video:
> https://www.youtube.com/watch?v=xMGhA5v4vxk
>
> This installation may take up to 2 hours, depending on your internet connection and computer speed.

Optional but recommended:

> If your computer does not have “git” version control software installed, download and install it from here:
> https://git-scm.com/downloads
>
> Here is the “How to” Video:
> https://www.youtube.com/watch?v=tc3Aoi5Z1FE

Optional but recommended:

> If you are on a windows computer, after installing git, you can install “TortoiseGit” to integrate the git capabilities into the windows File Explorer.
> Here is the download page:
> https://tortoisegit.org/download/
>
> Here is the “How to” Video with installation instructions and also examples on how to use git with your workshop repositories in this subject.
> https://www.youtube.com/watch?v=mSMGq3fTF-U

THE AT-HOME SECTION

1- Do one of the following two:

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

Go to following webpage:
https://github.com/Seneca-244200/OOP-SAB-Notes/blob/master/README.md
and watch this instructional video:
How to create a simple console application in Visual studio

Create a file called ide_steps.txt and in your own words write the steps to create a simple console application in visual studio.

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

Create a file called ide_steps.txt and in your own words write the steps to create a simple console application in the IDE of your choosing.

2- Open the in_lab section and modify your modules to include compilation safeguards:
Surround your prototypes in following code to prevent accidental multiple includeds as follows:

```
#ifndef SICT_HEADERFILENAME_H_
#define SICT_HEADERFILENAME_H_

Your header file content goes here

#endif
```

Here is the instructional video to see how compiler works and why you need the above in all your header files:
https://www.youtube.com/watch?v=EGak2R7QdHo

3- Have the two modules; "graph" and "tools" part of sict namespace.
4- Have the main in seneGraph use the sict namespace.

Note that from now on (to the end of semester) all your work must be implemented in the "sict" namespace and all your main modules should "use" it.

5- Create a file call reflect.txt and answer the following questions:

- What is a namespace? Explain its purpose.

- Why header files are needed? Explain.

AT_HOME SUBMISSION (CODE 30%, REFLECT AND IDE_STEPS 20%)

To test and demonstrate execution of your program use the same data as the in_lab section.

If not on matrix already, upload **ide_steps.txt, reflect.txt, tools.cpp, tools.h, graph.cpp, graph.h** and **seneGraph.cpp** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

```
~profname.proflastname/submit 244_w1_home <ENTER>
```

and follow the instructions.

Please note that a successful submission does not guarantee full credit for this workshop.

If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.