Part A – Introduction

Compiling Modules

Workshop 1 (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 period. If you attend the lab period and cannot complete the *in-lab* portion of the workshop during that period, ask your instructor for permission to complete the *in-lab* portion after the period. If you do not attend the workshop, you can submit the *in-lab* section along with your *at-home* section (see penalties below). The *at-home* portion of the lab is due on the day that is two days before your next scheduled workshop (23:59:59).

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

You are responsible to back up your work regularly.

LATE SUBMISSION PENALTIES:

- *In-lab* portion submitted late, with *at-home* portion: **0** for *in-lab*. Maximum of 7/10 for the entire workshop.
- If any of *in-lab*, *at-home* or *reflection* portions is missing, the mark for the workshop will be **0**/10.

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.

NOTE: 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.

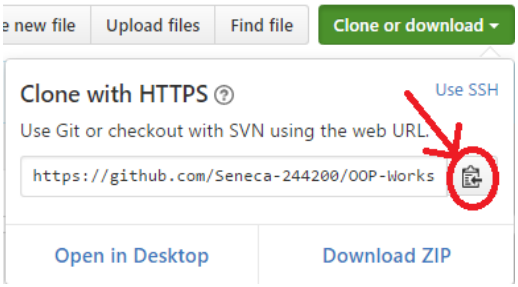
NOTE: `Tortoise Git` installation guidelines are in the *at-home* section of this lab.

RETRIEVE THE ORIGINAL PROGRAM:

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

NOTE: If the window 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:

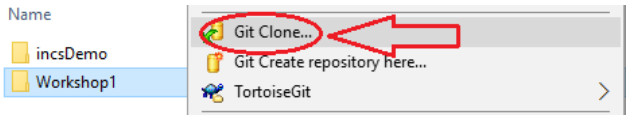


3. Open File Explorer on your computer and select or create a directory for your workshops.

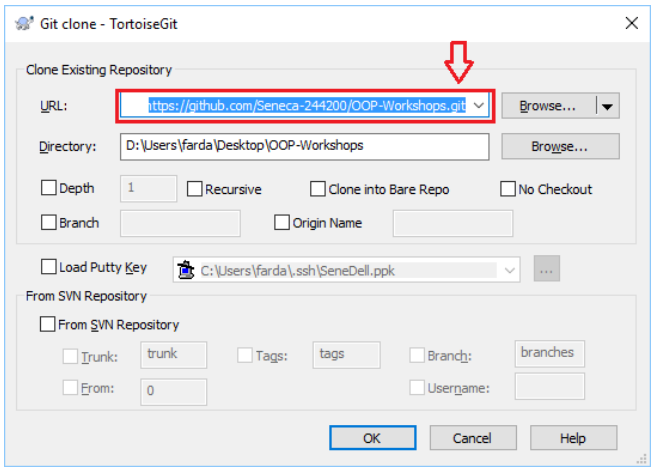
Now Clone (download) the original source code of `SeneGraph` (Workshop1) from GitHub in one of the following three ways: (methods 1 and 2 are preferred)

1. Using `TortoiseGit`:

   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 on your computer a clone (identical directory structure) of the directory on `Github`. Once you have cloned the directory, you can open the directory `OOP-Workshops/WS01` and start doing your workshop. Note that you will repeat this process for all workshops and milestones of your project in this subject.

IMPORTANT: 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 having to download it again. Note that this will only apply the changes made and will not affect any work that you have done on your workshop.

2. Using the command line:

   a. Open the `git` command line on your computer.
   b. Change the directory to your workshops directory.
   c. Issue the following command at the command prompt in your workshops directory:

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

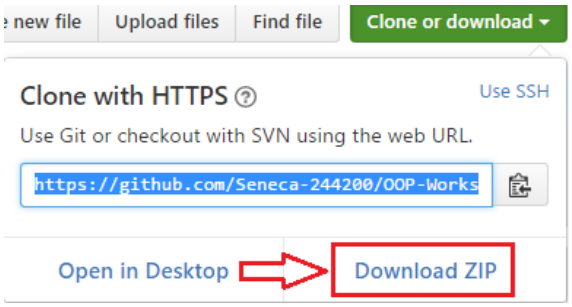
IMPORTANT: The URL for all the workshops are the same throughout the semester. The only thing that changes, is the workshop number.

This will create on your computer a clone (identical directory structure and content) of the `OOP-Workshops` directory on `Github`. Once you have cloned the directory, you can open subdirectory `OOP-Workshops/WS01` and start doing your workshop. Note that you will repeat this process for all workshops and milestones of your project in this subject.

IMPORTANT: 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 having to download it again. Note that this will only apply the changes made and will not affect any work that you have done on your workshop.

3. Using the "Download ZIP" option:

   a. Open https://github.com/Seneca-244200/OOP-Workshops and click on "Clone or download" button and click on "Download ZIP".



   b. This will download to your computer a zipped file copy of the workshop repository in `Github`. You can extract this file to where you want to do your workshop.

IMPORTANT: 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 (30%)

STEP 1: TEST THE PROGRAM

1. On Windows, using Visual Studio (VS)

    a. Open the `OOP-Workshops/WS01/in_lab` directory (that you just cloned or downloaded) and click on `in_lab.vcxproj`. This will open a Visual Studio (VS) project 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 your project.
    - Right click on "Source Files"
    - Select "Add/Existing Item"
    - Select `w1_in_lab.cpp` from the file browser
    - Click on "Ok"

    c. Compile and run the program by selecting "Debug/Start Without Debugging" or pressing "Ctrl-F5".

2. On Linux, in your `matrix` account.

    d. Upload `w1_in_lab.cpp` to your matrix account (Ideally to a designated directory for your oop244 workshop solutions). Then, enter the following command to compile the source file and create 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`

    e. Type the following to run and test the execution:

    `w1``<ENTER>`

STEP 2: CREATE THE MODULES

1. On Windows, using Visual Studio (VS)

In solution explorer, add three new modules to your project:

- `SeneGraph`
- `Graph`
- `Tools`

The `SeneGraph` module has an implementation (`.cpp`) file but no header file. The graph and tools modules have both implementation (`.cpp`) and header (`.h`) files:

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

Separate the functions in `w1_in_lab.cpp` and copy them into the modules as follows. Make sure that you include in the implementation file (`.cpp`) for each module `<iostream>` and insert the statement `using namespace std;` at the beginning of the file but after all include directives.

- The **`Tools`** module contains the `menu()` and `getInt()` functions. Copy their definitions to the module's `.cpp` file and their prototypes to the module's `.h` file. Make sure that you include `tools.h` in `tools.cpp`.

  To test that you have done this correctly, you can compile this module separately, by right clicking on `tools.cpp` and select `compile` from the menu. If the compilation is successful, most likely you have done it correctly.

  > **NOTE**: 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.

- The **`Graph`** module contains the `getSamples()`, `findMax()`, `printBar()` and `printGraph()` functions. Copy their definitions to the module's `.cpp` file and their prototypes to the module's `.h` file. Add the define statement for `MAX_NO_OF_SAMPLES` to `graph.h`. Make sure that you include `tools.h` and `graph.h` in `graph.cpp`.

  To test that you have done this correctly, you can compile this module separately, by right clicking on `graph.cpp` and select `compile` from the menu. If the compilation is successful, most likely you have done it correctly.

> **NOTE**: The equivalent of this on matrix is to add `-c` to the compile command:
>
> **`g++ graph.cpp –Wall -std=c++0x -c`**<ENTER>
>
> This will only compile `graph.cpp` and will not create an executable.

- The **SeneGraph** module contains the `main()` function. Make sure that you include `tools.h` and `graph.h` in `seneGraph.cpp`.

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 but only remove it).

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

2. On Linux, in your `matrix` account.

Upload the five files to your matrix account and compile the source code using the following command.

`g++ tools.cpp graph.cpp seneGraph.cpp -Wall -std=c++0x -o w1`<ENTER>

Run the program and make sure that everything works properly.

Output Sample (**red** values are entered by the 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 (30%)

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 that everything works properly.

Then, run the following script from your account: (replace `profname.proflastname` with your professor's Seneca userid)

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

and follow the instructions.

IMPORTANT: 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.

At-Home Section (30%)

Step 1: Prepare Your Computer

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

  Download Visual Studio Community from https://www.visualstudio.com/downloads/ and install it.

  At the start of installation, select the Universal Windows Platform Development and Desktop Development C++ options.

  This installation may take some time, depending on your internet connection and your computer's speed.

- Optional but recommended:

  If your computer does not have `git` version control software installed, download and install it from https://git-scm.com/downloads

  Here is the "how-to" video: https://www.youtube.com/watch?v=tc3Aoi5Z1FE

- Optional but recommended:

  On a Windows computer, after installing `git`, you can install `TortoiseGit`, which will integrate the `git` capabilities into Windows File Explorer.

  Here is the download page: https://tortoisegit.org/download/

  Here is the "how-to" video: https://www.youtube.com/watch?v=mSMGq3fTF-U. The video contains installation instructions along with examples on how to use `git` with your workshop repositories.

Step 2: Complete the Workshop

1. Open the *in-lab* section of your workshop solution and add compilation safeguards to your modules. Surround your prototypes in your header files with the following code to prevent multiple includes:

```
#ifndef NAMESPACE_HEADERFILENAME_H
#define NAMESPACE_HEADERFILENAME_H

// Your header file content goes here

#endif
```

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

2. Enclose the prototypes and function definitions in your `Graph` and `Tools` modules within the `sict` namespace.
3. Include in your `SeneGraph` module a directive to use the `sict` namespace.

Reflection Section (40%)

Create a text file named `reflect.txt` and briefly answer the following questions:

1. What is a namespace? Explain its purpose.
2. Why are header files needed? Explain.
3. In the instructions above, you were directed to compile only `.cpp` files but not the `.h` files. Explain why you should **never** compile header files.
4. Explain why you should **never** include `.cpp` files in another file.
5. Explain in your own words what have you learned on this workshop.

Quiz Reflection

Add a section to `reflect.txt` called "**Quiz X Reflection:**" (replace the '**X**' with the number of the last quiz).

Identify the questions from the quiz that you did not answer correctly.

Under each question, specify your mistake and provide the correct answer. If you missed the last quiz, enter all of the questions and the correct answer for each.

At-Home Submission

To test and demonstrate execution of your program use the same data as the *in-lab* section.

If not on matrix, upload `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 that everything is working properly.

Then, run the following script from your account (replace `profname.proflastname` with your professor's Seneca userid):

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

and follow the instructions.

IMPORTANT: 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.