Computations

Workshop 2 (out of 10 marks - 1% of your final grade)

In this workshop, you will code and execute a C-language program that accepts numerical values from the user, stores the values in variables of appropriate data type, performs calculations on the stored variables and casts from one data type to another.

LEARNING OUTCOMES

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

- to declare variables of integral and floating point types
- to code a simple calculation using C operators and expressions
- to accept a numerical value from the user using scanf
- to cast a value from one data type to another
- 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 (with a penalty; see below). The “at-home” portion of the lab is due on the day that is two days before your next scheduled workshop (23:59).

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 70/70 for at-home and reflection
- If any of in-lab, at-home or reflection portions is missing the mark will be zero.

ORIGINAL SOURCE CODE

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.

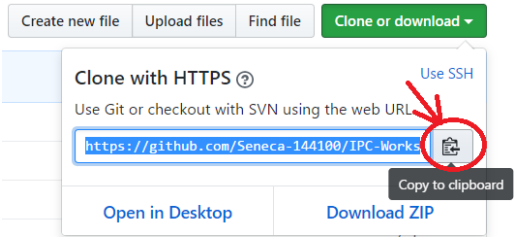
NOTE: `Tortoise Git` installation guidelines are in the *at-home* section of this lab.

Retrieve the original source code:

1. Open https://github.com/Seneca-144100/IPC-Workshops and click on "Clone or download" Button; this will open "Clone with HTTPS" window.

NOTE: If "Clone with SSH" is highlighted, then click on "Use HTTPS":

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

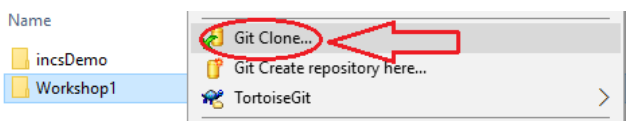


3. Open File Explorer on your computer and select or create a directory for your workshops.

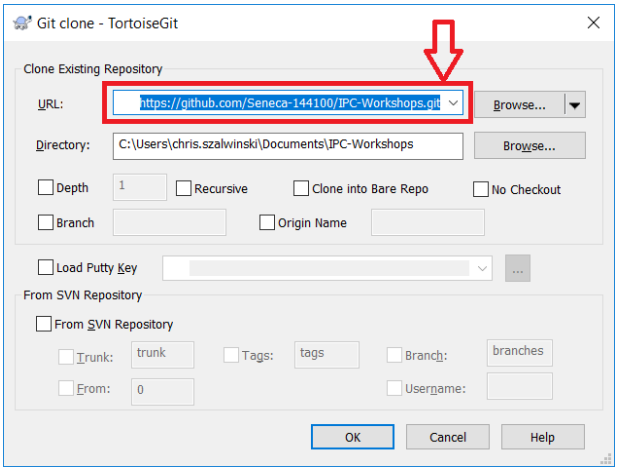
Now Clone (download) the original source code of the Workshop from GitHub in one of the following three ways: (methods 1 and 2 are preferred)

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 `IPC-Workshops/WS02` and start doing your workshop. Note that you will repeat this process for all subsequent 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-144100/IPC-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 directory on `Github`. Once you have cloned the directory, you can open the directory `IPC-Workshops/WS02` and start doing your workshop. Note that you will repeat this process for all subsequent workshops and milestones of your project in this subject.

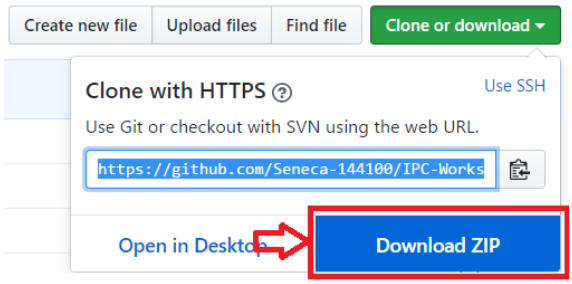
IMPORTANT: If your professor makes any changes to the workshop, you can issue the command `git pull<ENTER>` in the cloned repository directory to update and sync your local workshop to the one on `Github` without having to download it again. Note

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-144100/IPC-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: DONE IN CLASS (30%)

Download or clone workshop 2 (**WS02**) from https://github.com/Seneca-144100/IPC-Workshops

In the in_lab directory of workshop 2 click on in_lab.vsxproj to open the workshop in Visual Studio.

In the Visual Studio solution explorer, open and write your code in cashRegister.c for workshop 2.

Start the program by asking the user to enter the amount due. Print the following message:

```
Please enter the amount to be paid: $
```

Assume that the user enters 8.68, this is what the screen should look like:
(<ENTER> means hitting the enter key)

```
Please enter the amount to be paid: $8.68 <ENTER>
```

Read the amount due and store it as a double.

Calculate the number of loonies and quarters required to pay the amount due and display the following:

Loonies required: 8, balance owing $0.68.
Quarters required: 2, balance owing $0.18

Execution and Output Example:

```
Please enter the amount to be paid: $8.68
Loonies required: 8, balance owing $0.68
Quarters required: 2, balance owing $0.18
```

For submission instructions, see the SUBMISSION section below.

IN_LAB SUBMISSION:

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

If not on matrix already, upload your **`cashRegister.c`** to your matrix account. Compile and run your code and make sure everything works properly.

```
AtThePrompt> gcc -Wall cashRegister.c –o ws<ENTER>

-Wall activates the display of warnings in GCC compiler.
-o sets the executable name (ws)
```

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

```
~profname.proflastname/submit 144_w2_lab <ENTER>
```

and follow the instructions.

AT_HOME: (30%)

Install git and Tortoise Git:

- Optional but recommended:

  If your home 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 the 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.

Complete this Section:

After completing the in_lab section, edit and upgrade cashRegister.c to add the GST to the total entered by the user.

Display the amount of the GST and then the total due. Then display the number of quarters, dimes, nickels and pennies required to pay the total amount.

Problem: 8.68 * .13 is equal to 1.1284, which should be rounded up to 1.13, and the format specifier %.2lf will display as 1.13, but the number will be truncated to 1.12 on a C standard

compiler, such as matrix, if we multiply by 100 and cast to an int, which we must do at some point in the program.

To find the correct value for GST do the following:

GST = amountOwing * .13 + .005; (result is 1.1334 which will resolve to 1.13 when either rounded or truncated)

After Calculating the number of loonies required, subtract the number of loonies from the amountOwing, then cast the remaining decimal portion to an int. The subsequent operations must be done using integer division and modulus. Although the answer can be derived without using the modulus operator, failure to use modulus will result in a grade of 0 for the At_Home portion of the workshop.

Hint: read about casting, division and modulus in the notes

*Hint: .35 * 100 = 35*
You can output the value 5, stored in an int variable as $0.05 like this:
`printf("$%1.2lf", (float)intBalance/100);`

Execution and Output Example:

```
Please enter the amount to be paid: $8.68
GST: 1.13
Balance owing: $9.81
Loonies required: 9, balance owing $0.81
Quarters required: 3, balance owing $0.06
Dimes required: 0, balance owing $0.06
Nickels required: 1, balance owing $0.01
Pennies required: 1, balance owing $0.00
```

AT-HOME REFLECTION (40%)

Please provide brief answers to the following questions in a text file named **`reflect.txt`**.

1- What do you think is the most important thing you learned in this workshop?
2- Why must you add .005 to the value derived when calculating the GST owing?
3- What is the result of a casting operation?
4- Why do you think you were required to use the modulus operator in this workshop?

Note: when completing the workshop reflections it is a violation of academic policy to cut and paste content from the course notes or any other published source, or to copy the work of another student.

AT_HOME SUBMISSION:

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

If not on matrix already, upload your **cashRegister.c** and **reflect.txt** to your matrix account. Compile and run your code and make sure everything works properly.

```
AtThePrompt> gcc -Wall cashRegister.c –o ws <ENTER>
```

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

~profname.proflastname/submit 144_w2_home <ENTER>

and follow the instructions.