Computations

Workshop 2 (out of 10 marks - 2% 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.

Problem:

When a salesperson behind an old style cash register rings all the items for a customer in and finds out what is the total amount, she has find out what is the total tax, and then add it to the amount. Then she has to ask for the amount from the customer and give the change back.

Your task for this workshop is to:

1- Receive the total amount due.
2- Display the amount due after tax.
3- Ask the for the total cash received from the customer.
4- Display the change due back to the customer.
5- Finally show the salesperson the number of coins to give back the customer in Toonies, Loonies, Quarters, Dimes, Nickels and Pennies.

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 workshop is **due no later than four (4) days following the in-lab assigned date (even if that day is a holiday) by 11:59PM.**

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.

IN_LAB: DONE IN CLASS (30%)

Download or clone workshop 2 (**WS02**) from **https://github.com/Seneca-144100/IPC144SCP-Notes/tree/master/Workshops/WS02**

In the in_lab directory of workshop 2, right-click on the file in_lab.vsxproj and select the "Open With" context menu item then select "Microsoft Visual Studio 2017" to open the workshop in Visual Studio.

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

For the in-lab section we will display the change in Toonies and Loonies and experience the lack precision in double values.

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

Please enter the amount due: $

1- Assume the user enters 9.03, this is what the screen should look like:
(<ENTER> means hitting the enter key)

Please enter the amount due: $9.03<ENTER>

Read the amount due and store it as a double.

2- Calculate the amount after tax in a double variable and display the tax and total amount due:
Amount due after $1.17 tax is: $10.20

3- Ask the user to enter the amount of cash received from the customer:
Enter the total cash received from the customer: $20<ENTER>

4- Display the total change due back to the customer:
Total change due back to the customer: $9.80

Store the change back value in Cents in an integer value.
Using division find the number of Toonies (200 cents) and using modulus update the amount of change back in cents.

5- Display the results (for the change owning value, use casting so you can use `%.2lf` for formatting)
Toonies required: 4, change owing $1.79 <- Note that we lost a penny!

6- Repeat the exact same logic on change owing to find the number of loonies:
Loonies required: 1, change owing $0.79

Note: Although there are many ways to do this lab, you are required to use the modulus (%) operator.

Execution and Output Example:

```
Please enter the amount due: $9.03
Amount due after $1.17 tax is: $10.20
Enter the total cash received from the customer: $20
Total change due back to the customer: $9.80
Toonies required: 4, change owing $1.79
Loonies required: 1, change owing $0.79
```

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 and replace **sxx** with your section)

```
~profname.proflastname/submit 144SXX_w2_lab <ENTER>
```

and follow the instructions.

AT_HOME: (30%)

After completing the in_lab section, edit and upgrade **cashRegister.c** to fix the loosing penny in your calculation and continue breaking down the change owed to Quarters, Dimes and Pennies.

To fix the problem with double's precision go back to step four of in-lab:
before storing the amount of "change back" in Cents in an integer value add 0.5 to the cent value in double.
This will force the casting to result in the nearest integer to the double value (nearest high or nearest low):
234.2 + 0.5 = 234.7 cast to int: 234 therefor 234.2 → 234
234.6 + 0.5 = 235.1 cast to int: 235 therefor 234.6 → 235

After fixing the value, display the amount of the GST and then the total due. Then in addition to Toonies and Loonies, display the number of quarters, dimes, nickels and pennies required to pay the total amount.

Execution and Output Example:

```
Please enter the amount due: $9.03
Amount due after $1.17 tax is: $10.20
Enter the total cash received from the customer: $20
Total change due back to the customer: $9.80
Toonies required: 4, change owing $1.80
Loonies required: 1, change owing $0.80
```

```
Quarters required: 3, change owing $0.05
Dimes required: 0, change owing $0.05
Nickels required: 1
Pennies required: 0
```

AT-HOME REFLECTION (40%)

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

In three or more paragraphs, explain what you learned while doing this workshop. Tell us what was interesting and what you found difficult. Explain why it is often a best practice to convert floating-point values to integers when performing arithmetic operations. Why we lost a penny in the in-lab section? Why is it a best practice to use the modulus operator rather than division and subtraction to find a remainder?

Reflections will be graded on clarity of thought, grammar and spelling.

Note: when completing the workshop reflection 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 and replace **SXX** with your section)

~profname.proflastname/submit 144SXX_w2_home <ENTER>

and follow the instructions.