Computations

# Expressions

Workshop 2 (worth 3% of your final grade)

In this workshop, you will write a program that calculates the minimum number of coins required to pay an amount owing. The coins to be used are loonies, quarters, dimes, nickels and pennies.

## LEARNING OUTCOMES

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

- do simple calculation using C operators and expressions.
- use scanf to read a number from console
- cast values between integer and floating point

## SUBMISSION POLICY

Your workshops are divided in two sections; in_lab and at_home.

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, but there will be a reduction from 40% to 10% for the "in_lab" portion. The "at_home" portion of the lab is **due the day before your 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 for regularly backing up your work.

## IN_LAB: DONE IN CLASS (40%)

Download or clone workshop 2 from https://github.com/Seneca-144100/IPC-Workshop2

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.

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: (50%)

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.

*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 (10%)

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

1- What did you learn in this workshop?

## 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.

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.