Workshop #2

Worth: 2% of final grade

Breakdown: Part-1 Coding: 10% | Part-2 Coding: 40% | Part-2 Reflection: 50%

Introduction

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 the appropriate data type, performs calculations on the stored variables (including the modulus operator) and casts from one data type to another.

Topic(s)

- Computations: **Types**, **A Simple Calculation**, **Expressions**

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

- Part-1 is due on Thursday
- Part-2 is due on Sunday
- In each case, the due date is the **end of day by 23:59** EST (UTC – 5)
- **Late submissions will NOT be accepted**

All files you create or modify MUST contain the following 4 pieces of information; your:

1. Full name
2. Student Number
3. Seneca Email Address
4. Section Information Code

Notes

- Due dates are in effect **even during a holiday**
- You are responsible for **backing up your work regularly**

Late Submission/Incomplete Penalties

If any of Part-1, Part-2, or the Reflection portions are missing, the mark will be **ZERO**.

Part-1 (10%)

Instructions

Download or clone workshop 2 (**WS02**) from https://github.com/Seneca-144100/IPC-Workshops
Note: If you use the download option, make sure you **EXTRACT** the files from the .zip archive file

1. In the part1 directory of workshop 2, right-click on the file "**part1.vsxproj**" and select the "**Open With**" context menu item then select "***Microsoft Visual Studio 2019***" top open the workshop in Visual Studio.
2. In the Visual Studio solution explorer panel, expand the "Source" folder view and double-click the file "**w2p1.c**" and in the main editing window, write your code for workshop 2
3. Review the "Part-1 Output Example" (next section) to see how this program is expected to work
   Note: Canadian currency is comprised of 6-coin denominations as follows:

Coin Name	Value ($)	Value (cents)
Toonie	$2.00	200 cents
Loonie	$1.00	100 cents
Quarter	$0.25	25 cents
Dime	$0.10	10 cents
Nickel	$0.05	5 cents
Penny	$0.01	1 cent

4. Start the program by displaying to standard output (screen) the underlined title (2 lines).
5. Display the following message to the user:

```
Enter dollars and cents amount to convert to coins: $
```

6. Assume the user enters the value **9.92**, this is what the screen should look like:
   Note: <ENTER> represents hitting the enter key – this is not displayed

```
Enter dollars and cents amount to convert to coins: $9.92<ENTER>
```

7. Read from standard input (keyboard) the user entered value and store it to a variable of double type
8. Calculate the number of **toonies**, **loonies**, and **quarters** required to dispense the amount due and display the remaining owed with each coin denomination

Part-1 Output Example *(Note: Use this data for submission)*

```
Change Maker Machine
====================
Enter dollars and cents amount to convert to coins: $9.92

$2.00 Toonies  X 4 (remaining: $1.92)
$1.00 Loonies  X 1 (remaining: $0.92)
$0.25 Quarters X 3 (remaining: $0.17)

Machine error! Thank you for letting me keep $0.17!
```

Part-1 Submission

1. Upload (file transfer) your source file "**w2p1.c**" to your matrix account
2. Login to matrix in an SSH terminal and change directory to where you placed your workshop source code.
3. Manually compile and run your program to make sure everything works properly:
   ```
   gcc -Wall w2p1.c -o w2 <ENTER>
   ```
   If there are no error/warnings are generated, execute it: ***w2*** *<ENTER>*
4. Run the submission command below (replace **profname.proflastname** with **your professors** Seneca userid and replace **NAA** with your section):
   ```
   ~profName.proflastname/submit 144w2/NAA_p1 <ENTER>
   ```
5. Follow the on-screen submission instructions

Part-2 (40%)

Instructions

In a new source code file "**w2p2.c**", upgrade the solution to part-1 to include a service fee of 5% to be applied against the user entered value and complete each coin denomination until there is a zero remaining balance.

1. Review the "Part-2 Output Example" (next section) to see how this program is expected to work
2. You will need to display the service fee which is calculated as 5% of the entered value.
3. You will need to deduct the service fee from the entered value to determine the total amount to be dispensed in coins.
   Problem:
   9.92 * 0.05 = **0.496**, which should be rounded up to **0.50**. The printf format specifier "%.2lf" will apply rounding when displaying this value, **BUT don't be fooled**, the actual variable value will **NOT be rounded**.
4. You will need to apply appropriate rounding so when you multiply by 100 (to work in total cents) and cast to an int type, which we must do at some point in the program (see below), the value will properly represent the desired rounded value for any entered initial amount.
5. Part-2 of this workshop **requires you to use the modulus (%) operator (failure to use the modulus operator in all required cases will result in a zero grade)**.
   Hints:
   - Review the course notes about casting, division, and the modulus operator
   - The modulus operator only works with integral data types (int)

- You will need to work in total cents and use **integer division** with the **modulus** operator as required to properly calculate each coin denomination and respective remaining balance at each interval.
- All necessary calculations after determining the balance to be dispensed (less the service fee) must use **integer division** and the **modulus** operator (do NOT use a float or double type variable from this point onward in your program –use casting when needed)
- To display a floating point number based on an integer type variable holding 5 (representing 5 cents) as $0.05, you can do the following:

```
printf("$%1.2lf", (double)intBalance/100);
```

Part-2 Output Example *(Note: Use this data for submission)*

```
Change Maker Machine
====================
Enter dollars and cents amount to convert to coins: $9.92
Service fee (5.0 percent): 0.50
Balance to dispense: $9.42

$2.00 Toonies  X 4 (remaining: $1.42)
$1.00 Loonies  X 1 (remaining: $0.42)
$0.25 Quarters X 1 (remaining: $0.17)
$0.10 Dimes    X 1 (remaining: $0.07)
$0.05 Nickels  X 1 (remaining: $0.02)
$0.01 Pennies  X 2 (remaining: $0.00)

All coins dispensed!
```

Reflection (50%)

Instructions

- Create a text file named “reflect.txt”
- Record your answers in the reflect.txt file for each of the following:

1. Given the following C statements:

```
printf("10.10 = %.15lf\n", 10.10);
printf("10.12 = %.15lf\n", 10.12);
printf("10.15 = %.15lf\n", 10.15);
```

   Briefly explain why 10.12 doesn't display the value expected?

2. What are **all the possible values** that can be returned based on the following statements using the modulus operator (you many use a range to simplify your answer):

a) intValue % 2
b) intValue % 3
c) intValue % 100

3. Apply the modulus operator in the following situation:

   You are hosting a party and you order 11 pizzas. Each pizza has 12 slices, and the average person eats 3 slices, and you expect 21 people at the party. How many pizza's will be completely consumed and how many extra slices will be needed to feed the expected number of guests? What does the formula look like if you write it in the C language? Fill-in the missing parts (underscored):

```
int totalSlices = 11 * 12;
int estimatedSlicesConsumed = 21 * 3;
int wholePizzasEaten = ________________   _   ________________ ;
int extraSlicesNeeded =  ________________   _   ________________ ;
```

4. Briefly explain why it is often a best practice to convert floating-point values to integers when performing arithmetic operations?

Academic Integrity

It is a violation of academic policy to copy content from the course notes or any other published source (including websites, work from another student, or sharing your work with others).

Failure to adhere to this policy will result in the filing of a violation report to the Academic Integrity Committee.

Part-2 Submission

1. Upload your source file "**w2p2.c**" to your matrix account
2. Upload your reflection file "**reflect.txt**" to your matrix account (to the same directory)
3. Login to matrix using an SSH terminal and change directory to where you placed your workshop source code.
4. Manually compile and run your program to make sure everything works properly:

   ```
   gcc -Wall w2p2.c -o w2 <ENTER>
   ```

   If there are no error/warnings are generated, execute it: ***w2*** *<ENTER>*

5. Run the submission command below (replace **profname.proflastname** with **your professors** Seneca userid and replace **NAA** with your section):

   ```
   ~profName.proflastname/submit 144w2/NAA_p2 <ENTER>
   ```

6. Follow the on-screen submission instructions