CS 002 - Assignment 2: Cash Register

Collaboration Policy

We encourage collaboration on various activities such as lab, codelab, and textbook exercises. However, **no collaboration between students is allowed on the programming assignments**. Please be sure to read and understand our full policy at: Full Collaboration Policy

Submission Instructions

Submit to Hypergrade.

Assignment: Cash Register

Your goal is to simulate a cash register. Your program should first ask for the purchase amount and then the amount paid by the customer. Then the program should calculate the total change and the quantities of each monetary denomination to arrive at that change amount. Finally, the program should output the total change followed by how many of each type of currency to give back to the customer.

*Algorithm outline: Calculate the change in dollars, then convert it to pennies, assigning this number into a new variable. Then use **integer** division to calculate the number of each coin denomination, and use the **modulo** operator to update the remaining number of pennies each time.*

This requires that we convert from floating point data (change in dollars) to integer data (change in pennies), a process called *casting:* specifically, we use an operator called a static cast as follows:

```
int_variable = static_cast<int>( fp_variable );
```

We have to be careful about this: it doesn't always give us the result we expect!
Since it *truncates* (rather than rounding), a number such as 46.999999 (which, even as a double, displays as **47**) will become **46** as an int *(go ahead and try it!)*

To get around this, all we have to do is add **0.5** before casting: this will force the fp number to be ***rounded*** to the nearest int:

```
x = static_cast<int>( y * 100 + 0.5 );
```

Convince yourself that this does indeed work for *any* decimal fraction.
For this lab exercise, what are good variable names for x and y?
And what data types are they, respectively?

Run both of the below examples.

First utilize proper casting with the "rounding" amount as described above.
Then run both again without it. *See the difference?*

(There is an urban legend about a programmer for a bank who truncated all transactions, rather than rounding them, and routed all those fractions to his own account. He was supposed to have made quite a tidy sum before he was caught!)

Example Runs (inputs are **bold & underlined**)

```
Enter purchase amount: 18.89

Enter amount received: 20.00

Total Change: $1.11

dollars 1
quarters 0
dimes 1
nickels 0
pennies 1
```

```
Enter purchase amount: 25.56

Enter amount received: 40

Total Change: $14.44

dollars 14
quarters 1
dimes 1
nickels 1
pennies 4
```