Upload your source code to Canvas using `YourName_Assignment1.cpp` format.

At the beginning of the program, state the assignment number, the purpose of the assignment, the authorship information, and the date the assignment was last edited.

Use short comments throughout your code to document your program. Your code should be easy to read and self-guided.

Objective: Generate a **coin dispenser machine**. This machine would dispense change for any appropriate monetary value. This value should be less than $100. The machine uses an algorithm which *dispenses the least number of coins* based on the US coin system of **quarters**, **dimes**, **nickels,** and **pennies**.

As we talked about in class, $10.75 worth of coins would be 43 quarters (minimum number of coins).

$10.74 worth of coins would dispense 42 quarters, 2 dimes, 4 pennies.

You can also use the Canadian coin system which includes 2-dollar (200 cents) and 1-dollar coins, just indicate at the begin of your program which monetary system you're referring to. If you have any other monetary system in mind, feel free to use it. The algorithm would not change and only the coin names and values would be modified.

Instructions: Write a program that would prompt the user to input some value in dollars and cents in the format of **$xx.xx** and solve the equivalent number of coins. This number of coins can be at most $100.00.

For an input of $5.42 the machine dispenses the following:

- 21 quarters
- 1 dime
- 1 nickel
- 2 pennies

To do this, you can first convert the input amount into cents:
$xx.xx * 100 (cents)
(Alternatively, you can input $xx.xx as a string using the string library. You can then read each character and convert it back into a number like "$10.74" would be dissected as '1' , '0', …. Each character can be converted back to numeric value by subtracting it from '0' like '1' – '0' becomes number 1.)

You now have an integer number of pennies that need to be divided into 25, 10, 5 and single pennies. The coin designations are easily found by an algorithm the uses the two operators (/ for int division) and (% for remaining amount). The algorithm is shown below:

$5.42 is equivalent to 542 cents.

First decide on the number of *larger* coins, quarters.
542 / 25 ----> 21 quarters
542 % 25 ----> *17 cents of change*

Next, use the change to count the *next largest coin*, the dimes.
17 / 10 ---- > 1 dime
17 % 10 -----> *7 cents of change*

Use the change to count the nickels.
7 / 5 -----> 1 nickel
7 % 5 ---------> *2 cents change*

Finally, pennies are remaining, must be less than 5 pennies.
2/1 --------> 2 pennies
2%1 --------> 0 cents left

The next step is displaying the original dollar amount along with the coin designations and their number. The display should print something like follows:

Value: $5.42
Quarter: 21
Dime: 1
Nickel: 1
Penny: 2
Test your program on a variety of input values.

Note 1: If a negative amount is input, the program should reject it and abort. Use *exit(1)* function to abort. This function can be used by including the header *#include <cstdlib>* .

Note2: All the coins must be declared as CONST variables and used by their identifier names all throughout the program. For example:

```
CONST int QUARTER = 25;
//this variable is a constant and cannot be modified.
```

Identifiers of constant values should all be capitalized!

Note3: There may be some situations where your coin dispenser counts one less penny. For instance, $2.99 might dispense one less penny. Test your program so many times where you do find a discrepancy. Next decide why this is happening and what solution you are offering. The data types of double type typically lack accuracy especially during type conversions.

HAVE FUN!