Lab 2

Lab Overview and Objectives:
Make a program that takes a properly formatted amount of money and prints the most efficient change breakdown for that quantity.
Supported bills: $20, $10, $5, $1, $.25, $.1, $.05, $.01

Example:
Input: $29.87
Output:
1 - $20.00
0 - $10.00
1 - $5.00
4 - $1.00
3 - $0.25
1 - $0.10
0 - $0.05
2 - $0.01

Input: $15.32
Output:
0 - $20.00
1 - $10.00
1 - $5.00
0 - $1.00
1 - $0.25
0 - $0.10
1 - $0.05
2 - $0.01

Because you want the most efficient breakdown, you should never have so much of a denomination that it could be a larger bill. For example, 5 pennies is a nickel so you should never have more than 4 pennies. For this lab, “Input” will be a hard coded variable in main.

Prelab:
1. In your own words, describe what a variable is.
2. In your own words describe what #include is and does. Give an example.

Name A File lab2.c:
#include <stdio.h> /**< Include the stdio.h file to allow for printf() to be used*/
/** @brief Main Function
   * @param void This function does not accept any input variables
   * @return This function would return an error code to the OS if needed.
   */
int main(void) {
    float input = 29.87; /**< 'input' will hold the amount of money */
    // put your code in here
    return (0);
}

You will edit the lab2.c code. You must:
1. Indent all lines inside of main as above.
2. If you create new variables you must comment them as I have above.
   Start your in-line comment with /** and end it with */
   Describe briefly what the variable is used for.
3. Show your code and run your working code for the TA and submit your code to Lab2 on TEACH by the end of lab.
Possible Tools For This Program (and after this lab you are responsible for knowing):

**Integer division** – Integers cannot support decimal precision, division with integers will truncate the result.

\[
\begin{align*}
5/3 &= 1 & \text{3 goes into 5 1 time} \\
11/2 &= 5 & \text{2 goes into 11 5 times}
\end{align*}
\]

In general, 
\[
a/b = n \quad \text{assuming b*n + r = a (where r is the remainder and r < b)}
\]

**Modulus** – The mod operator (%) can be used to find the remainder from division.

Examples:

\[
\begin{align*}
5\%3 &= 2 & \text{because 2 is the remainder of 5/3} \\
10\%2 &= 0 & \text{because 0 is the remainder of 10/2} \\
6\%20 &= 6 & \text{because 6 is the remainder of 6/20}
\end{align*}
\]

In general, 
\[
a\%b = r \quad \text{assuming b*n + r = a (where r is the remainder and r < b)}
\]

**Casting** – Type casting allows variables to act like another type.

For example, if you have a float amount, you can type cast it to an int; doing so will truncate the decimal. We print float variables with %f.

```c
float amount = 59.89;
int truncatedAmount = (int)amount;
printf("amount is %f and (int)amount is %i\n", amount, truncatedAmount);
```

Prints: amount is 59.889999 and (int)amount is 59

**printf formatting techniques** – You can specify how many digits to the right of the decimal point to print. Did you notice that in the above example 59.89 was printed as 59.889999? If you are writing a program in dollars and cents, it doesn’t make sense to print out so many digits. Plus there was roundoff error because the floating point number 59.89 cannot be stored exactly in the computer’s memory.

Example:

```c
printf("amount is %5.2f\n", amount); // %5.2f says to print up to 5 digits with 2 to right of decimal pt
```

Prints: amount is 59.89

**Possible Algorithm:**

Suppose you start out with an input of $36.19

1. How many times does 20 go into the input? Obviously we want an integer here, not 1.8095. That’s how many 20 dollar bills you need. 1 twenty dollar bill.
2. How much money is left after you reduce input by the number of 20’s. In our example that is 36.19 – 20 = 16.19
3. Repeat steps 1 & 2 but for 10 and input is 16.19.
4. Repeat steps 1 & 2 but for 5 and a new input.
Another Possible Algorithm:
Suppose you start out with an input of $36.19. Let’s not deal with floating point numbers. Let’s multiply by 100 to get 3619.

1. How many times does 2000 go into the input? Obviously we want an integer here, not 1.8095. That’s how many 20 dollar bills you need. 1 twenty dollar bill.
2. What is the remainder after you divide by 2000?
   \[ 3619 \% 2000 = 1619 \]
3. Repeat steps 1 & 2 but for 1000 and the new input of 1619.
4. Repeat steps 1 & 2 but for 500 and the new input.
...

Study Questions (written answers due at the beginning of Lab3):

```c
int main(void) {
    char school[] = "Oregon State";
    char grade;
    int x = 20, y = 3;
    int a = x/y;
    int b = x%y;
    int c = y%x;

    float z = 21.74;
    printf("%i\n", (int)z);
    printf("%5.1f\n", z);
}
```

1. What are the values of a, b, and c?
2. What is printed?
3. Using table 24-2 in your book, write a single printf statement that prints the values of school, grade, x, and z.

Lab 2 Summary:

<table>
<thead>
<tr>
<th>TASK</th>
<th>Completed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prelab</td>
<td>4pts.</td>
</tr>
<tr>
<td>Correct indentation</td>
<td>2pts.</td>
</tr>
<tr>
<td>Commented all new variables</td>
<td>2pts.</td>
</tr>
<tr>
<td>Correct number of 20’s printed out</td>
<td>1pts.</td>
</tr>
<tr>
<td>Correct number of 10’s printed out</td>
<td>1pts.</td>
</tr>
<tr>
<td>Correct number of other bills / coins printed out</td>
<td>4pts.</td>
</tr>
<tr>
<td>Study Questions</td>
<td>4pts.</td>
</tr>
</tbody>
</table>
Extended Learning:
Modify your program to prompt the user to enter an amount instead of using what the programmer stored inside the variable input.
If the user enters a negative number, print an error message and return -1.
Do not print out bills or coins where you need 0 of them.
Hints: Research if statements in your book.
Read in a floating point number by using %f inside scanf.

Demonstrate your program to a TA at the beginning of Lab3.
Submit your code to Lab2Extra by the end of lab in week 3.