CSC 101 Programming Assignment 2: Enhancements to Making Change; Advanced Numeric Computations

ISSUED:	Friday, 6 April
DUE:	On or before 11:59:59PM Friday 20 April, via `handin` on unix1
POINTS POSSIBLE:	100
WEIGHT:	4% of total class grade
READING:	Textbook chapters 2 through 4

Overview

This assignment has two parts. Part A involves some enhancements to Programming Assignment 1 from last week. Part B is a new program involving geometric computations with real numbers.

Program Specification, Part A

Update the solution to Programming Assignment 1 as follows:

a. Read input and perform all computations using real numbers instead of integers. E.g., instead of inputing 216 in cents, input 2.16 in dollars.

b. Format the output so that dollar and cent values are output as real numbers, with exactly two decimal places, and preceded buy a dollar sign.

c. Accommodate larger purchases by adding three more denominations for change: five dollars, ten dollars, and twenty dollars.

d. Perform checking on all input values, disallowing any negative values. Upon input of the first negative value, the program outputs the error message "Inputs must not be negative." and terminates immediately without performing any further computation.

e. Perform checking on the amount tendered, disallowing it to be less than the amount of the purchase. If the tendered amount is less than the purchase amount, the program outputs the error message "The amount tendered is insufficient by X.", where X is the amount in dollars and cents that the tendered amount is less than the purchase amount. After the error output, the program terminates immediately without performing any further computation.

f. Format the output so the the amount of each denominations uses the singular form of the denomination if the value is 1, or the plural form otherwise. See the first sample run below for an example.

Here are some sample runs of the updated change making program:

Sample 1: Change back for a $100 dollar bill

```
Input the amount of the purchase, in dollars and cents: 22.16
Input the amount tendered, in dollars and cents: 100.00

Total change due = $77.84

Change in twenties through pennies is:
    3 twenties
    1 ten
    1 five
    2 dollars
    3 quarters
    0 dimes
    1 nickel
    4 pennies
```

Sample 2: Error message for negative purchase amount.

```
Input the amount of the purchase, in dollars and cents: -5
Inputs must not be negative.
```

Sample 3: Error message for negative tendered amount.

```
Input the amount of the purchase, in dollars and cents: 5
Input the amount tendered, in dollars and cents: -1
Inputs must not be negative.
```

Sample 4: Error message for insufficient funds.

```
Input the amount of the purchase, in dollars and cents: 105.00
Input the amount tendered, in dollars and cents: 100.00

The amount tendered is insufficient by $5.00.
```

Program Specification, Part B

Acme Storage Containers, Inc. specializes in storage containers "of all shapes and sizes". They rent the containers for use at construction sites or other locations where someone needs temporary storage. When a container is returned by the renter, it must be fumigated and repainted inside and out.

The available container shapes are rectangular, domed, and cylindrical. They are available in a range of sizes. In order to determine how much fumigant and paint to buy for a returned container, the volume and surface area of the container must be computed. Your program will perform these computations.

The program starts by inputting one character to designate the container shape: ’r’ for rectangular, ’d’ for domed, and ’c’ for cylindrical. Following the shape input, the program inputs two real numbers that specify the base and height dimensions of a container. For a rectangular container, the base is the length of one of the square sides of its floor; for the other two shapes, the base is the diameter of its floor. For all three shapes, the height measure is how tall the container is from floor to top.

The program outputs two real numbers for the volume and paintable surface areas for the selected container shape. The output is to three decimal points of precision. The *paintable* surface area is the total surface area both inside and out, *minus* the area of the bottom (one side) of the floor. For the purposes of these computations, you may assume that the thickness of the walls is negligible, so that both the inside and outside surface areas are the same.

For calculation purposes, the dome shape is defined as a *hemispheroid*. It has a circular base and a variable height. Its surface area and volume are defined with quadratic hemispheroidal formulae.

Here are three sample runs, one for each shape:

```
Input shape of container: r
Input the base and height, separated by spaces: 4 8.5
Volume: 136.000
Paintable surface area: 320.000

Input shape of container: d
Input the base and height, separated by spaces: 3.25 9
Volume: 49.775
Paintable surface area: 192.079

Input shape of container: c
Input the base and height, separated by spaces: 6.5 4.25
Volume: 141.028
Paintable surface area: 273.122
```

If the program user enters a shape character other than ’r’, ’d’, or ’c’, then the program outputs the error message "The shape must be r, d, or c." and terminates immediately without performing any further computation. If the user enters a value less than or equal to zero for one or both of the numeric inputs, then the program outputs the error message "The base and width must be positive values." and terminates immediately without performing any further computation.

Program Structure and Development

Develop two separate C programs, one for Part A and another for Part B. The programs must be named `make_change2.c` and `containers.c`. They must compile error free with the following `gcc` commands:

```
gcc -ansi -pedantic -Wall -Werror make_change2.c -o make_change2
gcc -ansi -pedantic -Wall -Werror containers.c -o containers
```

Here’s a special note if you are going to use the math library, and in particular the `round` library function from `<math.h>`. To compile on `unix1` you need to add a couple extra compile flags, as follows:

```
gcc -ansi -pedantic -Wall -Werror -lm -D_ISOC99_SOURCE make_change2.c -o make_change2
```

For both parts of the assignment, you must follow the following program structuring rule:

Variables may only be declared inside function bodies, not outside of functions.

For Part A of Program 1, you should start work from your solution to Programming Assignment 1. If you did not complete Assignment 1, you can use the instructor’s solution, in the file

```
101/solutions/programs/1/make_change.c
```

You can incrementally develop `make_change2.c` in the following steps:

1. get the program to work for real number input and output
2. get the program to work for the three new denominations of money; for this, you can add three new `get_change` functions, *OR* you could consolidate all of the change making into one function with the following signature:

   ```
   int get_amount(double change, int denomination);
   ```

3. add the error checking
4. add the proper formatting of output for singular versus plural

Be sure to update the documentation from Program 1 as necessary. E.g., since Program 2A uses real numbers instead of integers for input, the comments that describe the inputs should refer to real numbers instead of integers.

For Part B of the assignment, you must develop your solution by yourself, from scratch. Be sure to include all necessary documentation, i.e., top-level description, variable comments, and in-line code algorithm comments. The following is an incremental development strategy:

1. write a function to compute the volume and surface calculations for just a rectangular container; this function can also print the values to `stdout`
2. write the main function to input a base and height numbers, and then call the rectangular calculation function
3. write two more functions to calculate and print for domed and cylindrical shapes; you can test these functions independently by having `main` call just one at a time, and supplying test input values for just that type of container
4. update the main function to input a character for the shape, and a `switch` statement to call the appropriate calculation function; this will let you test all three functions from the same program
5. add error handling

Testing

As you develop, test your programs incrementally, as you add each program feature. A detailed test plan will be posted in the Program 2 testing directory.

Reading Resources

The C features necessary to complete this program are covered in chapters 2 through 4 of the book. You can find the formulae for computing volume and surfaces areas for the different shapes in a math book or web site that covers computational geometry.

It turns out that there a lot of variants online for the formulae of a dome. To avoid confusion, you can use the following:

```
volume = 2/3.0 * M_PI * pow(base/2, 2) * height
surface = 2 * M_PI * base/2 * height
```

This surface formula is the curved part of the dome, without the flat circular base. To get the total surface area of the dome, you need to add in the area of the circular base. And then remember from the specification above that the *paintable* surface area is the total surface area both inside and out, *minus* the area of the bottom (one side) of the floor.

Collaboration

NO collaboration is allowed on this assignment. Everyone must do their own individual work.

Program Turnin Procedure

Use the `handin` command on `unix1` as follows:

```
handin gfisher 101_prog2 make_change2.c containers.c
```