G51PRG Exercise One: Variables and Functions

Steven R. Bagley

Introduction

For this coursework, you are to implement a computer program which performs some simple calculations and prints the results to the screen. Since we have not yet explained how to provide input to a program, the values to be converted will need to be 'hard-coded' (i.e. explicitly put into the program). Subsequent exercises will teach you how to read input from other sources.

There are three parts to this exercise. In the first part, you will write a program that calculates the volume of a sphere with a given radius. The second part converts an amount of money in pounds into its largest note and coin constituents. Finally, the last part gets the computer to add two times together, which are specified in the 24hr clock notation.

Your finished program (or programs), when run, should execute each of the three sections in order, if you are unable to complete a section then credit will still be given for the work actually completed. When marking we will be looking at how your program is written as well as the output it produces – this, of course, means that you won't be able to cheat by using code such as this:

```
printf("The volume of a sphere with radius 5 is 523.5987750);
```

There is no single correct solution for this exercise; any program which correctly calculates the answers will be given a pass mark. To achieve higher grades, your program should make appropriate use of variables and functions in your solution. In particular, to obtain the top grades it is necessary to design your own functions and make correct use of them.

1. Volume of a sphere

Write a program which calculates the volume of a sphere, using the equation:

$$V = \frac{4}{3}\pi r^3$$

You should write a function `volumeForRadius()` that takes a `double` as a parameter and returns a `double` containing the result of calculating the above equation. You should take the value of π to be 3.14159265. Your program should call your function and calculate the volume of the sphere, for the following radiuses:

Radius (r)	
1	cm
4.657	cm
10	cm
42	cm

Your program should print 4 lines (one per radius), in the following format:

```
The volume of a sphere with radius 5 is 523.598775
```

2. Smallest money

Write a program that converts a monetary value (specfied in pounds) into its largest note and coin constituents. The denominations considered here will be £20, £10, and £5 notes, and £1 coins. For example, 47 pounds would break down into 2 × £20, 1 × £5, and 2 × £1. You will need to make sure that only `int`eger values of the different denominations are printed. For example, the program should print 4 lines (one per input value), in the following format:

```
£77 consists of 3 £20 notes, 1 £10 note, 1 £5 note, and 2 pound coins
```

In this case, it is perfectly acceptable to print out zeroes if no, say, £5 notes are needed. However, if you are feeling adventurous you might want to use an `if` statement to remove these and to make the code print 'note', 'notes', 'coin' and 'coins' as appropriate.

It is best to create a function to handle the conversion. You will not be able to return three values from the function, so you will need to call `printf()` from within the function. We will learn later in the module how we could return all three values from the function.

Your program should calculate the break down of currency for the four input values in this table:

Monetary value
£20
£79
£10
£42

Note: The £ can be a pain to print out correctly particular if you move your code from computer to computer. You will *not* be penalised if the pound sign doesn't print correctly.

Hint: This exercise requires you to use C's integer division operator and its modulus (%, i.e. the remainder) operator.

3. Adding Times

This section requires you to further practise integer arithmetic. In particular, C's modulus operator. In this section, you should add two times together that are encoded as `int`egers.

The first integer value represents a time of day on a 24 hour clock, so that the value 1055 represents "5 minutes to 11 in the morning", 1245 represents "quarter to one mid-day", and 2005 represents "5 minutes past 8 in the evening". To obtain the one integer value we have multiplied the hours by 100 and then added the minutes to it. The second integer represents a time duration in a similar way, so that 345 represents three hours and 45 minutes.

We wish to find the time at which an event starting at the time given by the first number, and lasting for the duration given by the second number, will end. Thus the duration is to be added to the first time, and the result printed out in the same notation. Thus if the input is "1245 345" the output should be 1630 which is the time 3 hours and 45 minutes after 12.45.

Note: There are a few extra marks for spotting that a start time of 2300 with a duration of 200, gives an end time of 100, not 2500. But don't worry if you can't get that working.

The program's output should be in the format:

```
Start time is 1245. Duration is 345. End time is 1630
```

and should calculate results for the following values:

Start Time	Duration
1045	345
800	435
2300	300

Hint: Again, this exercise requires you to use C's integer division and remainder (modulus) operator. It may help if you split the integers into hours and minutes first, then perform the calculations before combining them back into one value.