G51PRG Exercise One: Variables and Functions

Steven R. Bagley

Introduction

For this coursework, you are to implement a computer program in C, which performs some simple calculations and prints the result 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). Part of the next exercise will be to modify this program to read the input values from the user.

There are three parts to this program, and are arranged in increasing order of difficulty to implement. The first part calculates the amount of energy available from an object of a given mass using Einstein's famous equation. The second part converts a price in pennies into 'old money' (pounds, shillings and pence). Finally, the last part gets the computer to add two prices together — again specified in 'old money' — printing the result in pounds, shilling and pence. 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 be given for the work actually completed.

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:

```
printf("Energy for object of mass 45kg is 12345kJ\n");
```

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. Einstein's equation: $E = mc^2$

The first part of this exercise is to write a program that can calculate the relativistic energy of an object using Einstein's famous equation:

$$E = mc^2$$

You should write a function `energyForMass()` that takes a `double` as a parameter and returns the result of calculating the above equation. You should take the value of *c* to be `2.99792458`, otherwise the numbers will be very large. Using this value for *c* means that we can easily display the result in *petajoules* (10^{15} *joules*), by dividing the result by ten.

Your program should then call your function and calculate the amount of energy available (in *petajoules* — don't forget to divide the result by 10) for the objects of the following masses:

Mass	
1	kg
3.142	kg
10	kg
42	kg

and print the energy (one line per mass) calculated using the above equation in the form:

```
Energy available from object of mass 1kg is 0.8987552PJ.
```

2. Pounds, Shillings and Pence

The second section requires you to write a program that converts an amount of money specified in pennies into the old UK monetary units of pounds, shillings and pence. There are 12 pennies to the shilling, and 20 shillings to the pound. You will also need to make sure that only an `int`eger number of pounds and shillings and pence are printed. So, if the input value is 241 (pennies) then your program should print out:

```
241 pennies is 1 pounds, 0 shillings and 1 pennies
```

or for 42 pennies:

```
42 pennies is 0 pounds, 3 shillings and 6 pennies
```

Note: Do not worry about making your program switch between `shillings` and `shilling` etc. when appropriate.

Again it is probably best to create a function to handle the conversion, although you won't be able to return the three values from it so you'll need to `printf()` from within your function. We'll see in a later lecture how we could get return all three values.

Your program should calculate and printout the pounds, shillings and pence values for the following values:

Pennies
11
13
42
100

Hint: This exercise requires you to use C's integer division and remainder (modulus) operators.

3. Adding Prices Together

The final section follows on from the previous sections and again requires you to practise integer arithmetic. In this section, you should add two prices together and print the result. To make things tricky, the values are again specified in pounds, shillings and pence. You should take the two sets of values, add them together and print the result.

There are several ways to solve this problem — you could add the values individually and then adjust their values as appropriate, e.g. if you ended up with 13 pence, you could add change it to one pence and add one to the shillings result etc. Alternatively, you may find it easier to convert the prices to pence, do the calculation and then convert it back to pounds, shilling and pence. Both approaches are fine, and shows how there is often more than one way to solve a programming problem.

The program's output should be similar to:

```
    Adding £1 4s 3d and £2 17s 8d gives £4 1s 11d
```

and should calculate values for the following values

Price One	Price Two
£1 4s 3d	£2 17s 8d
£1 5s 9d	£0 1s 11d
£1 19s 11d	£0 0s 1d