Lab Exercise #3: C Formatted Input/Output

Getting started

Download lab materials from D2L (including this instruction and two starter codes)
Enter MimirIDE
Change into the cse220 directory
Create a new directory called lab03
Change into the new directory
Upload starter codes to MimirIDE, save them in /cse220/lab03/
Implement the program below in your lab03 directory

Program 1 Description

You are to write a program that converts weights from pounds to ounces and grams. Call your program **weightConversion.c**
Your program should ask the user to enter 4 weights, read the weights entered in the following format:

```
weight1/weight2/weight3/weight4
```

The program should then convert each weight from pounds to ounces and from pounds to grams and output the result as the following example:

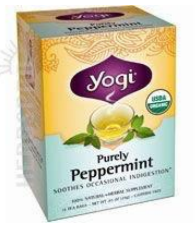
```
Weight 1:      2.50 lbs =      40.00 Oz =    1133.98 g
Weight 2:     15.00 lbs =     240.00 Oz =    6803.89 g
....
```

All values must be right justified, in a field of size 10, showing two digits after the decimal point.
Compile your program and generate an executable file called **weightConversion**.

Program 2 Description

For a number of years, manufacturers of goods sold in U.S. and Canadian stores have put a bar code on each product. This code, known as a Universal Product Code (UPC), identifies both the manufacturer and the product. Each bar code represents a 12-digit number, which is usually printed beneath the bars. For example, the following bar code comes from Yogi Purely Peppermint Tea.

0 76950 45047 9



The digits
0 76950 45047 9
appear underneath the bar code. The first digit identifies the type of item (0 or 7 for most items, 2 for items that must be weighted, 3 for dugs an health-related merchandise, and 5 for coupons). The first group of 5 digits identifies the manufacturer (76950 is the code for the Yogi company). The second group of 5 digits identifies the product (in this case Purely Peppermint Tea). The final digit is a "check digit", whose only purpose is to identify an error in the preceding digits. If the UPC is incorrectly scanned, the first 11 digits likely won't be consistent with the last digit, and the store's scanner will reject the entire code.

The final "check digit" is calculated with the following steps:

1. Add the first, third, fifth, seventh, ninth, and eleventh digits.
2. Add the second, fourth, sixth, eigth, and tenth digits.
3. Multiply the first sum by 3 and add it to the second sum.
4. Subtract 1 from the total.
5. Compute the remainder when the adjusted total is divided by 10.
6. Subtract the remainder from 9.

For the example UPC above:

1. First Sum = 0 + 6 + 5 + 4 + 0 + 7 = 22
2. Second Sum = 7 + 9 + 0 + 5 + 4 = 25
3. Total = (22 * 3) + 25 = 66 + 25 = 91
4. Adjusted Total = 91 – 1 = 90
5. Remainder = 90 % 10 = 0 (% is the modulus operator in C it divides the two numbers and gives the remainder)
6. Check Digit = 9 – 0 = 9

You need to write a program (named upc.c) that asks the user for a UPC and calculates the correct check digit. To avoid confusion, you will prompt (ask) the user to input the numbers in the groups they appear within on the barcode.

Example:

```
Enter the first (single) digit:
0
Enter the first group of five digits:
76950
Enter the second group of five digits:
45047
Check digit: 9
```

Note: the underlined characters are user input that was typed into the program. The rest of the text should be outputted by the upc program.

Hints:

- You should use scanf with a %1d placeholder to capture a single digit (or repeatedly to capture many single digits).
- The "%" operator takes the number on the left and divides by the number on the right, yielding the remainder.
    - Examples:
        - 10 % 3 is 1
        - 22 % 10 is 2

You should compile and run your program. Demonstrate that it works to the TA with some other UPC that you have on your person.