Computer Programming: Skills and Concepts
Tutorial 1 (Tue 3 Oct – Fri 6 Oct)

Decimalisation, and C syntax

Before 1971 British currency was divided into pounds, shillings and pence (in decreasing order of value). There were 20 shillings per pound, and 12 pence per shilling. The modern system was introduced in 1971 (the conversion was called "decimalisation") where the pound retained its value, but was now divided into 100 new pence (and the shilling was removed entirely from the new currency). This is the current system.

(i) Your task is to first consider the problem of converting from the "old money" into today's representation. Try to design your own program for solving this problem.

(ii) Next, consider the following buggy code:

```
#include <stdio.h>
#include <stdlib.h>

const OLD_PENCE_PER_SHILLING = 12;
const OLD_PENCE_PER_POUND = 240;
const NEW_PENCE_PER_POUND = 100;

int main(void)  {
  int pounds; shillings; oldpence; newpence;

  pounds = 4
  shillings = 7
  oldpence = 8

  oldpence = oldpence + shillings * OLD_PENCE_PER_SHILLING;
  newpence = ( oldpence / OLD_PENCE_PER_POUND ) * NEW_PENCE_PER_POUND;

  printf(%d %d/%d in old money , pounds, shillings, oldpence);
  printf(is %d.%d in new money.\n, pounds, newpence);
  return EXIT_SUCCESS;
}
```

What mistakes can you find? Correct the code!

Variables

What is printed on the screen for the following code fragments?

```
int a = 2; printf("%d", a);
int a = 3, b = 5; a = b/3 + a; printf("%d, %d",a,b);
int x,y; x = 3; y = x % 2; printf("%d,%d",x,y);
int x, y; y = 3; y = x + 2;  printf("%d,%d",x,y);
float a = 2.5; printf("%f", a);
float a = 2.5; printf("%d", a);
int a = 2.5 * (float) 5; printf("%d", a);
```

Printing large `float` or `double` variables

On slides 10, 11 of Lecture 4, we gave an example showing the difference in how `float`s and `double`s are initialised in a C program, and how they appear when printed out. The C program was:

```
#include <stdlib.h>
#include <stdio.h>

int main(void) {
  float x, x2;
  double y, y2;
  x = 6e5;
  x2 = -0.2223;
  y = 5e-8;
  y2 = -6e306;
  printf("Two floats are %f\n and %f.\n", x, x2);
  printf("Two doubles are %2.9f\n and %f\n", y, y2);
  return EXIT_SUCCESS;
}
```

The integer following `e` denotes the power (of 10) to multiply by.

The second printf gave an output like this:

```
Two doubles are 0.000000050
 and -60000000000000004151464359452186995442947633620854598420126115503945248
87240456918741880815778392846311318941394518041571623614758275072994875068520
76765339123136457002148018714284214841530693316940432073342282766995128786796
34094905773013933547655429167101887147924700636668768497796837912298082360151
24480
```

Notice that the output of second `double` variable prints a garbled string of digits, although the definition of the variable indicates a huge (306) number of 0s ... What is the explanation for this mystery?

scanf and Conditionals

What does the following code do?

```
int eggs;
const int EGGBOX = 6;

scanf("%d",&eggs);
if ( eggs % EGGBOX == 0 ) {
  printf("You have %d boxes of eggs\n", eggs/EGGBOX);
} else {
  printf("You need %d more eggs to make up %d boxes\n",
         EGGBOX - (eggs % EGGBOX), eggs/EGGBOX + 1);
}
```

Questions from the Labs

By now, you should all have had the opportunity to get up-to-date with the material of the 2nd Lab sheet (the conversion programs, and `whatday.c`). Any questions for your tutor on this material?