Notes on CP Tutorial Sheet for week 3 (2013/14):

The first question concerns decimalisation (converting from pounds, shillings, pence into the current system) as a computational task, and also finding the syntax and semantic errors in a mangled version of a program for doing the conversion.

(i) The question asks the students to first attempt the problem by themselves. Spend some time discussing their proposed solutions. They may raise the issue of using printf and scanf to read pounds, shillings, pence from the user and hence to solve the problem for general inputs - rather than the fixed values considered in the given program.

(ii) Now they must find the many errors in the given code.

It has the following syntax errors:
* missing int in the constant declarations
* semi-colons instead of commas between the variable declarations;
* missing semi-colons after the first three assignments;
* missing " round the format strings in the printfs

In addition, it has semantic errors
* bad rounding
* bad printing of new pence.
* One unintentional (when this program appeared in lectures) error. Oldpence is modified to include the contributions from shillings, but then the output still uses oldpence, instead of the original value of oldpence.)
* An additional error is putting the division and multiplication the other way round, and so falling in to the usual integer division trap.

The second question is simple integer arithmetic, and (a tiny bit) exploration of how the C compiler works in the absence of explicit casting.

The program tutw3b.c is for you to run to check all these things yourselves (if you want). Answers below.

You might want to use the 6th example (output X) to mention that floats are represented as sign(1 bit) exponent (7 bits) mantissa (24 bits). This is totally different to int .. so when you try to push a float variable through a %d channel just get junk. THIS IS a TOTALLY different reason for 'JUNK' in answer 4 (due to lack of initialisation).

Answers are:

```
2
4, 5
3,1
X,Y             where X is an arbitrary unknown integer, Y is X + 2
2.500000
X               X is just JUNK, trying to output a float via int format.
12              The float mult gets done BEFORE fitting to int var.
```

The program tutw3c.c is the program containing the code discussed in this Q (is floats.c on the course webpage-students can download).

Basic Explanation:
The initial explanation is that the "rules" for double variables are that they store floating-point values with a high exponent (up to about 10^306 or 10^307) ****to a precision of 15 digits***.
And if the students study the number output for y2, they will see that yes, definitely it is correct up to those 15 significant digits (16 to be exact).
I already discussed this simple explanation in the lecture.

Detailed Explanation:
They will wonder why there are lots of non-0 digits after the 15 guaranteed correct significant digits ... the expectation of an average person would be that after significant digits stop being guaranteed, that then the default would be to just have 0s afterwards ...

Need to point out that a *double* number is stored in (composite) *binary* representation
b_1 b_2....b_12 b_13.............b_64
where b_1 is for the *sign*, and
b_2 .... b_12 are for the exponent (2^11 = 2048 values,
about 1024 values of each sign +/-, note log_10(1024) ~ 3)
b_13 ... b_64 are for the significant digits...

The *point* (in regard to detailed explanation) is that the structure of the storage is binary-based, hence any "rounding" is based on this pattern
Hence those later digits in our huge double - are nothing to do with the way we defined our initial number (or how we think about numbers in our decimal world)

NOTE: for this question, there was a TYPO in the original tutorial sheet handed out in lectures (I assigned the value -6w306 to y2, that should have been -6e306).

The final program illustrates conditionals and scanf. It reads a number (of eggs) from the input. If the number fits exactly into N eggboxs, it tells them so, otherwise it tells them how many more eggs they need to fill N+1 eggboxes exactly.

Please encourage the students to raise any questions that came from the 2nd lab. It is especially likely they may have questions about "whatday.c", which is quite a challenging Q for new students.