Lab 2

COMP9021, Session 1, 2014

The purpose of this lab is to

- further practice some programming techniques we have already seen and which are not too problematic: tests, loops, operations on integers, and use of arrays;
- work under the constraint of modifying or completing some existing code;
- come up with simple solutions, that might not be the first ones that come to mind.

Do not forget to write clear comments for any aspect of your code that requires explanation.

1 A number that satisfies a given number of constraints

Fifteen people make a statement about a number N. Two of these people are wrong, the other are right. Here are the statements.

1. N is a multiple of 2;
2. N is a multiple of 3;
3. N is a multiple of 4;
4. N is a multiple of 5;
5. N is a multiple of 6;
6. N is a multiple of 7;
7. N is a multiple of 8;
8. N is a multiple of 9;
9. N is a multiple of 10;
10. N is smaller than 1000;
11. N is smaller than 750;
12. N is smaller than 550;
13. N is smaller than 500;
14. N is greater than 400;
15. N is greater than 450.

Download the following stub available as `question1.c`, open it with Emacs (as usual, typing `C-c o`) or your preferred editor, change the comment as indicated, and complete the program so that it finds and outputs the value of N.

2 ☞ Finding special triples of the form $(n, n+1, n+2)$

Write a program that finds all triples of consecutive positive three-digit integers each of which is the sum of two squares, that is, all triples of the form $(n, n+1, n+2)$ such that:

- n, $n+1$ and $n+2$ are integers at least equal to 100 and at most equal to 999;
- each of n, $n+1$ and $n+2$ is of the form $a^2 + b^2$.

Hint: As we are not constrained by memory space for this problem, and as we have some experience with integers but not with floating point numbers, we might declare an array that can store an `int` for all numbers n in $[100, 999]$, equal to 1 in case n is the sum of two squares, and to 0 otherwise.

Do not forget to use `#define` to give evocative names to special values (in particular, the minimum and maximum values that a variable can take in a loop).

3 ☞ Finding the largest prime factor of a number

Write a program that finds the largest prime factor of a positive number provided as input, as prompted by the program. Your program should behave as follows.

```
$ a.out
Enter a strictly positive number: 15
The largest divisor of 15 is 5.
$ a.out
Enter a strictly positive number: 17
The largest divisor of 17 is 17.
$ a.out
Enter a strictly positive number: 2420
The largest divisor of 2420 is 11.
```

We assume that the input is "not too large" so that it can fit in an `int` (we will see later what "not too large" actually means).

Hint: This can be done *without* first generating prime numbers smaller than the input.