Notes/Remarks on Handout 2 and Exercise Sheet 2

1) Euclidean Algorithm

OBJECTIVE: Suppose we are given two numbers, a & b. we would like to find their largest common factor.

QUESTION: Is there a standard way we can find out what their largest common factor is? Who knows – maybe a & b are really large and complicated numbers, and it's really hard to guess what their largest common factor is just by looking at them.

SOLUTION: The Euclidean Algorithm helps us with this!

Roughly speaking, what the `while` loop of algorithm does is that it takes a pair of numbers (e.g. a & b) and it spits out a "smaller" pair of numbers[1] (e.g. a' & b') that have the same largest common factor as the initial pair of numbers. More succinctly, we have that lcf (a,b) = lcf (a', b').

Since (a',b') are a "smaller" pair of numbers than (a,b), it should be easier to figure out what their largest common factor is, except maybe they are still too complicated for us to figure out. So, we plug (a' & b') into the `while` loop again, and it spits out an even smaller pair of numbers (e.g. a'' & b'') which have the same largest common factor as our inputs, i.e. lcf (a',b') = lcf (a'', b''), and we continue until we get a pair of numbers where it is "obvious" what their largest common factor is[2].

HOWEVER: This is all very nice, except how do we know this algorithm actually works? More precisely, how do we know that `while` loop actually takes in a pair of numbers (e.g. a & b) and spits out a pair of numbers (e.g. a' & b') that indeed have the same largest common factor?

Realise that this question is basically asking whether the largest common factor changes or not during the `while` loop, i.e. whether the largest common factor is a loop invariant of the Euclidean algorithm. A proof of this fact is found in your lecture notes, page 11 of Handout 2.

Indeed, one of the main reasons behind introducing this example is to motivate why we are interested in loop invariants.

---

[1] In particular, just going by the pseudo-code, note that $a'=b$ and $b'= a \bmod b$. Generally speaking, it is clear that $a \bmod b$ (which is the remainder we obtain by dividing a by b) is less than a.

[2] In particular, suppose we have a pair of numbers, x & y, where y divides x. This means that: (i) the remainder $r = 0$, where $r = x \bmod y$; (ii) the largest common factor of x and y is y itself.

2) Exercise 2.1

Recall: In order to prove some statement is a *loop invariant*, we have to check that the statement

- Holds before the `while` loop is begun
- Holds after the body `B` has been executed *provided* it was true before the execution of the body, i.e. if it's true for the N^{th} execution of the loop, it is also true after the $(N+1)^{th}$ execution.

The main difficulty of this exercise is:

(1) Knowing exactly what we are given, and what we have to prove;
(2) Keeping track of variables whose values keep changing. In other words, it helps for us to be very organized in our thinking.

For this reason, it may be helpful to have a visual representation of what's going on. Here's a table of all the assignments of the variables at various stages of the algorithm – we fill in the values simply by following the pseudo-code:

	x	y	u_x	v_x	u_y	v_y	u	v
Before `while` loop	a	b	1	0	0	1		
N^{th} loop	x^{old}	y^{old}	u_x^{old}	v_x^{old}	u_y^{old}	v_y^{old}		
$(N+1)^{th}$ loop	y^{old}	r	u_y^{old}	v_y^{old}	$u - k^* u_y^{old}$	$v - k^* v_y^{old}$	u_x^{old}	v_x^{old}

TASK: We want to prove that the following two statements are loop invariants:

- $x = u_x * a + v_x * b$
- $y = u_y * a + v_y * b$

SOLUTION:

First , let us check that these two statements hold before the `while` loop. Reading off the values of the table, this amounts to verifying that:

- $x = 1 * a + 0 * b = a$
- $y = 0 * a + 1 * b = b$

which we know to be true, because of how we assigned the initial values to x and y.

Next, let us assume that the two statements hold after the N^{th} loop. We *want to show* that this implies that the statements hold after the $(N + 1)^{th}$ loop as well.

Let us be more explicit. We said that we assume the two statements hold after the N^{th} loop – but what does it mean for us to assume that the following statements

- $x = u_x * a + v_x * b$
- $y = u_y * a + v_y * b$

to be true for the N^{th} loop? Well, reading off the values of the table, this means we are **assuming** the following two statements a) and b) are **true**:

a) $x^{old} = u_x^{old} * a + v_x^{old} * b$
b) $y^{old} = u_y^{old} * a + v_y^{old} * b$

And we would like to **use** this information **to prove** that the two statements are true for the $(N+1)^{th}$ loop. But what exactly does it mean for the two statements to be true for the $(N+1)^{th}$ loop? Well, reading the values off the table again, it means that we **want to prove** that the following two statements are true:

1) $y^{old} = u_y^{old} * a + v_y^{old} * b$
2) $r = (u - k* u_y^{old})* a + (v-k* v_y^{old})* b$

Well, we know that statement 1) is true because we already assumed that b) is true. So we get that one for free. Awesome!

But what about statement 2)? Well, by Theorem 1 of Handout 1, we know that $x^{old} = k*y^{old} + r$, which we can rearrange to get:

$$r = x^{old} - k*y^{old}$$

Now, by statements a) and b), we know what the values of x^{old} and y^{old} are, so we can plug them into the expression to get:

$$\begin{aligned} r &= u_x^{old} * a + v_x^{old} * b - k*(u_y^{old} * a + v_y^{old} * b) \\ &= u_x^{old} * a + v_x^{old} * b - k*(u_y^{old} * a) + k*(v_y^{old} * b) \\ &= (u_x^{old} - k* u_y^{old})* a + (v_x^{old} - k* v_y^{old})* b \\ &= (u - k* u_y^{old})* a + (v-k* v_y^{old})* b \end{aligned}$$

where the last equality follows from reading off the values of u and v from the table, once again. Now, this computation proves that

$$r = (u - k* u_y^{old})* a + (v-k* v_y^{old})* b$$

Which is exactly statement 2), as desired!

In sum, we assumed statements a) and b) were true, and we showed that this assumption implies that statements 1) and 2) are true. In other words, we showed that if the two statements hold for the N^{th} loop, they hold for the $(N+1)^{th}$ loop as well. Putting everything together, we proved that the two statements are in fact loop invariants!