Exercises for Computability and Complexity, Spring 2017, Sheet 4

Please return next Thursday, March 2, in class.

Note. All problems on this sheet are about showing that certain functions are primitive recursive. This is done by checking that the function in question can be "built up" by using the rules 1. – 5. from Definition 5.1 in the lecture notes. When you use one of these rules for an argument in your proof, please always specify the arities of the functions that you construct.

Problem 1 (easy) Show that the function *plus2*: $\mathbb{N} \to \mathbb{N}$, $plus2(n) = n + 2$, is primitive recursive.

Problem 2 (a little less easy, but has a super short solution) Show that the function *minus1*: $\mathbb{N} \to \mathbb{N}$, $minus1(n) = \max(0, n - 1)$ is primitive recursive. Hint: there is a very compact way of doing this, exploiting the fact that the primitive recursion scheme condition $h(n + 1, x) = g(n, h(n, x), x)$ has the "minus 1" operation already built in in the first arguments n +1 and n passed to h and g.

Problem 3 (medium) Show that the function *evensquare*: $\mathbb{N} \to \mathbb{N}$, defined by $evensquare(n) = n$ if n is uneven, else $evensquare(n) = n^2$, is primitive recursive. You may assume that *square*: $\mathbb{N} \to \mathbb{N}$, $square(n) = n^2$, is primitive recursive. Hint: you may find it helpful to construct *evensquare* from a number of helper functions which you construct before assembling *evensquare*.