Solutions: Exam: Models of Computation TDA183 – DIT310

1. Prove or disprove the following statements:

   (a) There is a function f : Bool -> Bool in Haskell (or some other programming language) with the property that f x = True if x terminates and f x = False if x does not terminate.
   This is the extensional halting problem, a proof can be found in the lecture notes

   (b) There is a program f in lambda-calculus which has a normal form under one computation strategy and has no normal form under another strategy.
   Let w be a program which has no normal form, for instance w = (\x.xx)(\x.xx). Then we can define f by f = (\y.x)w. This has no normal form if we start to compute the argument w, but has the normal form x if we use normal order evaluation.

   (c) The set of functions N -> N is enumerable.
   This is not true and the proof is a diagonalization proof found in the lecture notes

   (d) If we fully evaluate a program in X which has a weak head normal form then the evaluation terminates.
   The program (s loop), where loop is the program (rec x=x) is on weak head normal form, but the full evaluation does not terminate, since loop does not terminate

2. What does it mean that a function f : N -> N is Turing-computable?
   This is found in the lecture notes.

3. Explain how to use a fixpoint operator to define a recursive function!
   Suppose that we have a recursive function f defined by

   f = … f .... f … f …

   We can always rewrite this as:

   f = E(f)

   (by letting E be \x. … x … x … x …)

   This expresses that f is a fixpoint to E, and hence we can use a fixpoint operator to compute this: fix E will be the required fixpoint of E.

4. Give an example of a computable function (not using Ackermann's function) which cannot be expressed in PRF. Explain why!
   The function f which is everywhere undefined is computable (for instance by the program f x = f x in Haskell) but it is not definable in PRF, since all functions in PRF are total.