Exercise sheet 3: Kleene's Recursion Theorems

Exercise 1 (Undecidability via reduction)**.** *Using the reduction technique, show that the set* $\{n \in \mathbb{N} \mid \mathrm{dom}\,(\varphi_n) = \emptyset\}$ *is not decidable.*

Exercise 2 (Are there always fixed points?)**.** *Consider the total computable function* $f : \mathbb{N} \to \mathbb{N}$ *defined as* $f(n) = n + 1$*. Does this function have a fixed point in the sense of Kleene's First Recursion Theorem?*

Exercise 3 (Quine as a fixed point)**.** *How can the Kleene's First Recursion Theorem be directly used to prove the existence of a Turing machine* $\mathcal{T}$ *that upon receiving any input, halts and outputs* $[\mathcal{T}]$*?*

Exercise 4 (Self-Reference)**.** *Using the Kleene's Second Recursion Theorem, show that there exists a Turing machine* $\mathcal{T}$ *with input alphabet* $S = \{0, 1, |, (,)\}$ *that defines the following computational process on input* $x \in S^*$*:*

1. *Obtain your own description* $[\mathcal{T}]$
2. *Compute the concatenation* $x||[\mathcal{T}]$ *of* x *and* $[\mathcal{T}]$*.*
3. *If the length of* $x||[\mathcal{T}]$ *is less than 100 symbols, output 0.*
4. *Otherwise, output 1.*

Exercise 5 (Recursion)**.** *Most programming languages support recursion by allowing a function to call itself from within its own code. We illustrate this in the language of Turing machines on two examples. Using Kleene's Second Recursion Theorem, show that the following are indeed valid definitions of Turing machines.*

(a) *Defining the Fibonacci sequence (in a terribly inefficient way). We define a Turing machine* $\mathcal{T}$ *on input* $n \in \mathbb{N}$ *as follows:*

   1. *If* $n = 0$ *output 1.*
   2. *If* $n = 1$ *output 1.*
   3. *Otherwise, output* $\mathcal{T}(n-1) + \mathcal{T}(n-2)$*.*

(b) *Defining a potentially "bottomless" recursion is also possible. We define a Turing machine* $\mathcal{T}$ *on input* $n \in \mathbb{N}$ *as follows:*

   1. *If* $n = 0$ *or* $n = 1$ *output 1.*
   2. *If* n *is even, compute* $\mathcal{T}(\frac{n}{2})$*.*
   3. *Otherwise, if* n *is odd, compute* $\mathcal{T}(3n + 1)$*.*

Exercise 6 (Universal Quine [*])**.** *Use the construction outlined in the proof of the quine existence to write a quine in your favourite programming language.*

[*]This exercise is optional.