Models of Computation II, Exercises 3: TM & Comp. Functions

1. Consider the Turing machine $M = (Q, \Sigma, s, \delta)$ where $Q = \{start, skip1s, shift, append, skip{*}s, tidyup, finalise\}$, $\Sigma = \{\llcorner\!\lrcorner, 0, 1, *\}$ and the transition function $\delta \in (Q \times \Sigma) \rightharpoonup (Q \times \Sigma \times \{L, R\})$ is given by:

δ	$\llcorner\!\lrcorner$	0	1	$*$
$start$	$(start, \llcorner\!\lrcorner, R)$	$(skip1s, 0, R)$		
$skip1s$	$(skip{*}s, *, R)$		$(skip1s, 1, R)$	
$shift$		$(append, 0, R)$	$(append, 1, R)$	$(shift, *, L)$
$append$				$(skip{*}s, 1, R)$
$skip{*}s$	$(skip{*}s, *, R)$	$(tidyup, \llcorner\!\lrcorner, L)$	$(shift, *, L)$	$(skip{*}s, *, R)$
$tidyup$			$(finalise, 1, R)$	$(tidyup, \llcorner\!\lrcorner, L)$
$finalise$	$(finalise, 0, L)$			

   (a) Give the computation that M performs with the initial configuration $(start, \epsilon, 011\llcorner\!\lrcorner111\llcorner\!\lrcorner110)$. When working with pen and paper, you may wish to abbreviate the states $\{start, skip1s, shift, append, skip{*}s, tidyup, finalise\}$ as $\{st, s1, sh, a, s{*}, t, f\}$.

   (b) If M is given a tape containing a list of numbers encoded using the representation on Slide 11 of your lecture notes, then what function does it compute?

2. Turing machines can work with binary representations of numbers. We will represent numbers "backwards" – as bit strings with the least significant bit first. For example $25 = 2^0 + 2^3 + 2^4$ would be represented as 10011. (It could also be represented as 100110, 1001100, etc..)

   Define a Turing machine that increments a number in this representation.

3. The kind of register machines we have seen in the course are generally called Minksy machines (after their creator). We can also define other types of register machines, such as the Successor machine, which has three types of instruction:

`clear` R L	which sets register R to zero, and jumps to instruction L.
`increment` R L	which increments the value in R, and jumps to L.
`test` R_1 R_2 L_1 L_2	which tests whether the values in R_1 and R_2 are equal to each other. If they are, it jumps to L_1, if not it jumps to L_2.

   (a) Show that every Minsky machine has a corresponding Successor machine that computes the same function, and vice-versa. Hint: It is enough to implement the behaviour of the building blocks of Minsky machines using Successor machines and vice-versa. We **don't** expect you to prove your implementations correct.

(b) Explain why this supports the Church-Turing thesis.

(c) Explain why this means that the halting problem for successor machines is undecidable.

4. (a) Just as with register machines, we can code Turing machines as numbers. Explain why the set of numbers corresponding to the Turing machines that eventually halt when run on the empty input is undecidable.

   (b) There is an algorithm (implementable with a Turing machine) that, given the code of a Turing machine T produces a sentence F_T of first-order logic which is *satisfiable* (*i.e.* true in some structure) if and only if the machine T never halts when run on the empty input. What does this tell you about the set of satisfiable first-order sentences?

   (c) Gödel's completeness theorem states that a sentence of first-order logic is *valid* (*i.e.* true in every structure) if and only if it is *provable.* It turns out that, sentences and proofs in first-order logic can be represented as natural numbers in such a way that:

   - Every sentence and proof has a unique representation.
   - A Turing machine can check whether a pair of numbers corresponds to a sentence and a proof of that sentence.

   We say that a set is *semidecidable* (or recursively enumerable) if there is a Turing machine (or equivalently a register machine) that will always halt if the input value belongs to the set, and run forever otherwise.

   Is the set of valid first-order sentences semidecidable? Is it decidable? Why?

5. Explain why it is impossible to have a perfect virus scanner.