MA/CSSE 474 HW 14 problems (highlighted problems are the ones to turn in)

Be sure to read the numerous questions and answers in the main assignment document

17.1a
(#1) 6

17.1b
(#2)

17.3a
(#3) 9

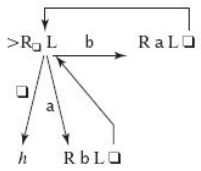
17.3b-d
(#4)

17.4
(#5) 9

17.6
(#6) 3

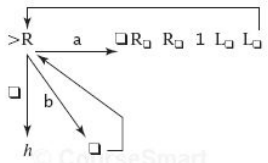
1. Give a short English description of what each of these Turing machines does:

a. $\Sigma_M = \{a, b\}$. $M =$



17.1a Don't just describe in English the *individual steps* the machine makes. Your answer should be a global one: in particular, describe how the final tape content is different from the original tape content.

b. $\Sigma_M = \{a, b\}$. $M =$



3. Construct a standard, deterministic, one-tape Turing machine M to compute each of the following functions:

   a. The function sub_3, which is defined as follows:

   $$sub_3(n) = \begin{array}{l} n - 3 \text{ if } n > 2 \\ 0 \text{ if } n \leq 2. \end{array}$$

   Specifically, compute sub_3 of a natural number represented in binary. For example, on input `10111`, M should output `10100`. On input `11101`, M should output `11010`. (*Hint*: You may want to define a subroutine.)

   b. Addition of two binary natural numbers (as described in Example 17.13). Specifically, given the input string $<x>;<y>$, where $<x>$ is the binary encoding of a natural number x and $<y>$ is the binary encoding of a natural number y, M should output $<z>$, where z is the binary encoding of $x + y$. For example, on input `101;11`, M should output `1000`.

   c. Multiplication of two unary numbers. Specifically, given the input string $<x>;<y>$, where $<x>$ is the unary encoding of a natural number x and $<y>$ is the unary encoding of a natural number y, M should output $<z>$, where z is the unary encoding of xy. For example, on input `111;1111`, M should output `111111111111`.

   d. The proper subtraction function *monus*, which is defined as follows:

   $$monus(n, m) = \begin{array}{ll} n - m & \text{if } n > m \\ 0 & \text{if } n \leq m. \end{array}$$

   Specifically, compute *monus* of two natural numbers represented in binary. For example, on input `101;11`, M should output `10`. On input `11;101`, M should output `0`.

4. Construct a Turing machine M that computes the function f: $\{a, b\}^* \rightarrow N$, where:

   $$f(x) = \text{the unary encoding of } max(\#_a(x), \#_b(x)).$$

   For example, on input `aaaabb`, M should output `1111`. M may use more than one tape. It is not necessary to write the exact transition function for M. Describe it in clear English.

6. Let M be a three-tape Turing machine with $\Sigma = \{a, b, c\}$ and $\Gamma = \{a, b, c, ␣, 1, 2\}$. We want to build an equivalent one-tape Turing machine M' using the technique described in Section 17.3.1. How many symbols must there be in Γ'?

17.11
(#7) 6-3

17.12
(#8) 6-6-6-6

17.13
(#9) 3

11. Prove rigorously that the set of regular languages is a *proper* subset of D.

12. In this question, we explore the equivalence between function computation and language recognition as performed by Turing machines. For simplicity, we will consider only functions from the nonnegative integers to the nonnegative integers (both encoded in binary). But the ideas of these questions apply to any computable function. We'll start with the following definition:

- Define the *graph* of a function f to be the set of all strings of the form $[x, f(x)]$, where x is the binary encoding of a nonnegative integer, and $f(x)$ is the binary encoding of the result of applying f to x.

For example, the graph of the function *succ* is the set $\{[0,1],[1,10],[10,11],\ldots\}$.

a. Describe in clear English an algorithm that, given a Turing machine M that computes f, constructs a Turing machine M' that decides the language L that contains exactly the graph of f.

b. Describe in clear English an algorithm that, given a Turing machine M that decides the language L that contains the graph of some function f, constructs a Turing machine M' that computes f.

c. A function is said to be partial if it may be undefined for some arguments. If we extend the ideas of this exercise to partial functions, then we do not require that the Turing machine that computes f halt if it is given some input x for which $f(x)$ is undefined. Then L (the graph language for f), will contain entries of the form $[x, f(x)]$ for only those values of x for which f is defined. In that case, it may not be possible to decide L, but it will be possible to semidecide it. Do your constructions for parts (a) and (b) work if the function f is partial? If not, explain how you could modify them so they will work correctly. By "work", we mean:

- For part (a): Given a Turing machine that computes $f(x)$ for all values on which f is defined, build a Turing machine that semidecides the language L that contains exactly the graph of f;
- For part (b): Given a Turing machine that semidecides the graph language of f (and thus accepts all strings of the form $[x, f(x)]$ when $f(x)$ is defined), build a Turing machine that computes f.

13. What is the minimum number of tapes required to implement a universal Turing machine?

Note on 17.12a:

L = { [<x>, <f(x)>], x∈ℕ}, where <x> means "the binary encoding of x" and <f(x)> means "the binary encoding of f(x)"

17.12b,c: Do these constructions for a general function-computing TM, not specifically for the successor function.

(c) You might find the concept of "dovetailing" helpful for this problem. If you have not seen that technique before, this reference will probably help:

http://lambda-the-ultimate.org/node/322