♢ **Best before:** September 26

1. Let $\Sigma = \{0, 1\}$.

   (a) Design a TM M_1 with input alphabet Σ that satisfies the following condition.
   There exist strings $x, y, z \in \Sigma^*$, such that
   - $|x| = |y| = |z| = 3$ (i.e., each of x, y, z has length 3),
   - M_1 accepts x, M_1 rejects y, and M_1 loops on z.

   Try to use as few states as possible.

   (b) Determine the partial function computed by your M_1 from part (a).
   See question 4 below for definition of *partial function*.

   (c) For this part, we treat the input and output of TMs to be natural numbers encoded in binary. Design TMs M_2 and M_3, both with input and output alphabet Σ, so that
   M_2 computes the function $f(n) = n - 1$ and M_3 computes the function $f(n) = n + 1$.

   You should use the TM simulator provided by Mustafa Quraish (former C63 student and TA) at
   `https://mustafaquraish.github.io/TMSim/`

2. A *Start-at-End TM* (SETM) is like a standard TM except for how it is initialized. Instead of the head being initially at the first cell, it is at the first blank cell (just past the end of the input).

   (a) Find an example of a standard TM and design an equivalent SETM (i.e., one that accepts exactly the same set of strings).

   (b) Prove that every standard TM has an equivalent SETM and vice versa.

3. Show that the collection of recognizable languages is closed under the operations of
   (a) union, (b) concatenation, (c) star, (d) intersection, (e) homomorphism.
   (See problem 1.66 on page 93 of Sipser for definition of homomorphism.)
   Do these closure properties also apply to co-recognizable languages?

4. Just as languages and decision problems are different ways of expressing the same concept, so are functions and non-decision problems (do you see how?). This question illustrates how Turing machines can be viewed as solutions to non-decision problems (as well as decision problems).

   (a) We define a function $f : \Sigma^* \to \Sigma^*$ to be *computable* if some Turing machine, on every input w, halts with just $f(w)$ on its tape.[1]

   For a function f, we define the language L_f to be $\{w\#y |\ w \in \Sigma^* \text{ and } y = f(w)\}$, where $\# \notin \Sigma$.
   Prove that a function f is computable if and only if L_f is decidable.[2]

   (b) Further to the ideas introduced in part (a), we introduce the notion of a *partial function*, which can be undefined for some values of its input. For example, $\ln x$ is undefined when $x = 0$. Formally we use a special symbol, $\perp$, to mean "undefined". For example, we would write $\ln 0 = \perp$.

   We extend the ideas from part (a) by adjusting our definitions.

[1] We can further extend the idea of computability to functions mapping from and to arbitrary sets as long as the elements of these sets can be encoded as finite strings from some alphabet. For example, we say the function $f : \mathbb{N}^2 \to \mathbb{N}$ defined by $f(x, y) = x + y$ is computable because there is a Turing machine that takes input $\langle x, y \rangle$ (encoding of x, y) and halts with just $\langle x + y \rangle$ (encoding of $x + y$) on its tape.

[2] When writing pseudocode, we can express the idea of a Turing machine returning a string x by writing "return x".

A partial function $f : \Sigma^* \to \Sigma^*$ is *computable* if for some Turing machine M, on every input w, M halts with just $f(w)$ on its tape whenever $f(w) \neq \perp$, and M loops whenever $f(w) = \perp$. Note that this definition of computability has the same meaning as the one in part (a) when f is defined on all of Σ^*.

A partial function $f : \Sigma^* \to \Sigma^*$ is a *total function* if it is defined on all of Σ^*. According to our definitions, a total function is also a partial function but a partial function is not necessarily a total function.

For a partial function f, we define the language L_f by $L_f = \{w\#f(w) |\ w \in \Sigma^* \text{ and } f(w) \neq \perp\}$. Note that this definition of L_f has the same meaning as the one in part (a) when f is defined on all of Σ^*.

How does the computability of a partial function f relate to the language L_f?

This is a somewhat open question. Think about what properties L_f has when f is computable.

5. Show that a language is decidable iff some enumerator enumerates the language in shortlex order. See page 14 of Sipser for definition of *shortlex order*.

6. Do problem 3.22 on page 190 of Sipser (about decidability and life on Mars).

7. A *No-Left* TM (NLTM) is like a regular TM except that on each transition its head can only move one cell to the right or stay put at the same cell.

   (a) Consider the problem of determining whether a given NLTM halts on a given input. Formulate this problem as a language and prove that it is decidable.

   (b) Let M be an NLTM. Prove that $L(M)$ is decidable.

8. Let Σ be some alphabet and let $A \subseteq \Sigma^*$. Let $f : \Sigma^* \to \Sigma^*$ be a partial computable function. We define two languages $f(A)$ and $f^{-1}(A)$, both over alphabet Σ, as follows.

   $$f(A) = \{f(x) |\ x \in A\} \text{ and } f^{-1}(A) = \{x |\ f(x) \in A\}.$$

   (a) Prove that if A is recognizable, then so is $f(A)$.

   (b) Prove that if A is recognizable, then so is $f^{-1}(A)$.

   (c) Are the converses of parts (a) and (b) also true? Justify your answers.