Problem 1 A string $x$ is a substring of a string $y$ if there are some (possibly empty) strings $u$ and $v$ such that $y = uxv$. For example, 000 is a substring of 01001, but not of 00100; 101 is a substring of 101; and the empty string is a substring of 111. As before, $x^R$ denotes the symbol-by-symbol reverse of a string $x$.

a. Show that the following language is context-free but not regular:

$$L_a = \{ x^R \# y \mid x, y \in \{0, 1\}^* \text{ and } x \text{ is a substring of } y \}.$$

b. Show that the following language is not context-free:

$$L_b = \{ x \# y \mid x, y \in \{0, 1\}^* \text{ and } x \text{ is a substring of } y \}.$$

As examples, note that 01#0011 is in $L_b$ and 10#0011 is in $L_a$, but not vice versa. The alphabet for both language $L_a$ and language $L_b$ is $\Sigma = \{0, 1, \#\}$.

Problem 2 The input to a Turing machine is always a string, but we will want to use Turing machines to reason not just about strings but about objects such as graphs, automata, grammars, and even other Turing machines. To do this, we will encode each object $O$ that is the input to the Turing machine as a string $\langle O \rangle$ over the machine’s input alphabet $\Sigma$; the input to the Turing machine will consist of the object in encoded (i.e., string) form.

Explain how you would encode a DFA $A = (\Sigma_A, Q_A, \delta_A, q_0A, F_A)$ as a string $\langle A \rangle$ suitable for input to a Turing machine. You can choose anything convenient for the Turing machine’s alphabet $\Sigma$.

Problem 3 As we defined it in class, a Turing machine both writes to its tape and moves left or right with each transition. The output of the transition function includes both an element of $\Gamma$, specifying the symbol to write to the tape, and an element of the set $\{L, R\}$, specifying a direction to move the head.

In this problem, we consider a more limited Turing machine that can either write to its tape or move its head, but not both, in each transition. For these machines, the output of the machine will include an element of $(\Gamma \cup \{\perp\})$, with $\perp$ a special symbol meaning “do not write to tape,” as well as an element of $\{L, R, \perp\}$, with $\perp$ a special symbol meaning “do not move the head.” All together:

$$\delta: (Q \setminus \{q_a, q_r\}) \times \Gamma \to Q \times (\Gamma \cup \{\perp\}) \times \{L, R, \perp\}.$$
We require that the output of the transition function have \( \perp \) either in place of a symbol to write or in place of a direction to move, but not both — the machine must either write to tape or move its head.

a. Turing machines that can either write or move are described using the same 7-tuple as the ordinary Turing machines discussed in class and in Sipser, except for the modification to \( \delta \) given above. The start configurations and accepting configurations are the same as for ordinary Turing machines. However, the rules for how a configuration \( C_i \) yields a configuration \( C_{i+1} \) are different.

Give the rules for how \( C_i \) yields \( C_{i+1} \) for a Turing machine that can either write or move, including the special-case rules that apply when the machine’s read head is close to either edge of the area it has used so far on the tape.

b. Explain how to simulate a Turing machine that can either write or move (but not both at once) on an ordinary Turing machine (that can both write and move with each transition).

c. Explain how to simulate an ordinary Turing machine (that can both write and move with each transition) on a Turing machine that can either write or move (but not both at once).