CSE 105 - Theory of Computation (Winter 2026)
Homework 5: Turing Machines
Due Wednesday, February 18, 11:59pm

Instructions: There are two parts to this homework:

- Concept check (Question 1): Every student must complete this individually on Gradescope.
- Written Homework (Question 2-4): You may work individually or in a team of up to 3 people. Please ensure your name(s) and PID(s) are clearly visible on the first page of your submission, and then upload the PDF to Gradescope. If working in a group, submit only one submission per group: one partner uploads the submission through their Gradescope account and then adds the other group member to the Gradescope submission by selecting their name in the "Add Group Members" dialog box. You will need to re-add your group member every time you resubmit a new version of your assignment.

It is highly recommended (though not required) that you type your answers. It is your responsibility to make any handwriting clear and legible for grading. A LaTeX template for the homework is provided on Canvas. For ease of grading, please start each new problem on a separate page.

We will only be grading some of the problems below for correctness. However, because all of the concepts are important, we will not reveal which problems are being graded for correctness until after the assignment has been submitted. The remaining problems will be graded for completeness (i.e., does it look like there was a good-faith effort to solve the problem?).

Reading and extra practice problems: Chapter 3. Chapter 3 exercises 3.1, 3.2.

Problems:

1. **Concept check**

   Complete the assignment "Homework 5 - Concept Check" on Gradescope.

2. **Describing Turing machines**

   Recall that for $K \subseteq \Sigma^*$, we define the language

   $$\textsc{Substring}(K) := \{w \in \Sigma^* \mid \text{there exist } a, b \in \Sigma^* \text{ such that } awb \in K\}.$$

   Let $\Sigma = \{0, 1, 2\}$. In this question, you will give three separate descriptions of a Turing machine which recognizes the language $\textsc{Substring}(\{0^n1^n2^n \mid n \geq 0\})$. This may seem like a somewhat tedious process, but we think that it is important to see all the different descriptions in action at least once for a single language.

   Hint: First give a simplified description of the language $\textsc{Substring}(\{0^n1^n2^n \mid n \geq 0\})$.

(a) High-level description: description of algorithm (precise sequence of instructions), without implementation details of machine. Your description can use data structures and refer to specific positions in the input strings (without specifying memory management).

(b) Implementation-level description: English prose that describes the Turing machine head movements relative to contents of tape, and conditions for accepting / rejecting based on those contents.

(c) Formal definition: Give the 7-tuple of parameters

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

describing the Turing Machine. Represent the transition function δ by drawing the state diagram of the Turing machine. You may use the following conventions: omit the reject state from the diagram; any missing transitions in the state diagram are assumed to go to the reject state.

3. **This Turing Machine is broken... or is it?**

Let's consider a variant of a Turing Machine which models computation where the data keeps getting corrupted. A Corrupted Turing Machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ has some of the usual Turing Machine components: set of states (Q); input alphabet (Σ); transition function (δ); start state (q_0); accepting state (q_{acc}), rejecting state (q_{rej}). Unlike a normal Turing Machine, the tape alphabet $\Gamma \supseteq \Sigma$ has two (rather than one) special characters $\sqcup$ and $\lightning$:

- $\sqcup \in \Gamma$, $\sqcup \notin \Sigma$: This is the usual blank symbol.
- $\lightning \in \Gamma$, $\lightning \notin \Sigma$: This symbol indicates a cell that has been corrupted. The machine cannot write over any corrupted cell; that is, for each $q \in Q$,

$$\delta(q, \lightning) \in \{(r, \lightning, D) \mid r \in Q, D \in \{L, R\}\}$$

Computation in the Corrupted Turing Machine proceeds as normal except that sometimes when writing a tape symbol as intended, a $\lightning$ symbol is written instead. Thankfully, the pattern of corruption is predictable: the first write is corrupted, and then every other write thereafter is corrupted. That is, the first, third, fifth,... etc. writes are corrupted.

(a) Prove that for every **regular** language L, there exists a Corrupted Turing Machine M that recognizes L.

(b) It will turn out that the Corrupted Turing Machine is no less powerful than our usual definition of a Turing Machine. Let's break the proof into a few steps. First, define the alphabet $\Sigma_{\text{pairs}} := \Sigma \times \Sigma = \{(a, b) \mid a \in \Sigma, b \in \Sigma\}$ of pairs of symbols in Σ. Give the construction of a Corrupted Turing Machine that takes input $a = a_1 a_2 \cdots a_n \in \Sigma^*$ and rewrites it as pairs of symbols interspersed by corrupted symbols. To be precise, the Corrupted Turing Machine should take starting tape configuration

a_1	a_2	$\ldots$	a_{n-1}	a_n	$\sqcup$	$\sqcup$	$\ldots$

to the tape configuration

↯	(a_1, a_2)	↯	(a_3, a_4)	↯	...	↯	(a_{n-1}, a_n)	↯	␣	␣	...

with the head once again pointing to the first cell. For simplicity, you may assume that n is even.

(c) Starting from the tape configuration we created in the previous step, describe the implementation of a Corrupted Turing Machine which simulates the computation of any normal/uncorrupted Turing Machine.

Challenge; not graded: A key idea in the construction above was to increase the number of symbols in our tape alphabet. Can you do the same construction *without* increasing the size of the tape alphabet? That is, for every language L recognized by a normal Turing machine over alphabet Σ and tape alphabet $\Sigma \cup \{\text{␣}\}$, is there a Corrupted Turing Machine with tape alphabet $\Sigma \cup \{\text{␣}, \text{↯}\}$ that recognizes L?

4. **True/False enumerator**

For each of the following statements, determine if it is true or false. Clearly label your choice by starting your solution with **True** or **False** and then provide a brief (3-4 sentences or so) justification for your answer.

(a) Every enumerator enumerates an infinite language.

(b) Let E be any enumerator and M be any Turing machine. If $L(E) = L(M)$, then M enters the reject state for all strings not in $L(E)$.

(c) Let E be any enumerator over Σ. Suppose $a, b \in \Sigma^*$ and $a, b \in L(E)$. If E prints a before b, then $|a| \leq |b|$.

(d) Let M decide language L over Σ such that M halts on all inputs $w \in \Sigma^*$ in $|w|^{2023}$ steps. There exists an enumerator E with the following properties: $L(E) = L(M)$; and if $a, b \in L$ and $|a| < |b|$, then E prints a before b.

(e) Let N be a nondeterministic Turing machine. There is an enumerator E that enumerates the set of all and only strings accepted by N that have odd length.