

# Homework 5

CSE 105, Fall 2025

Due: Monday November 10, 11:59pm

Your assignments in this class will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should always explain how you arrived at your conclusions, using mathematically sound reasoning, where applicable. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

## Instructions:

- Read each question carefully.
- Selection of questions
  - You are required to answer one question from each topic in this homework.
  - You may choose any one question from the set of questions within each topic.
- Each question is worth 5 points (20 points total).
- Submission guidelines:
  - Upload a single file to Gradescope for each group.
  - All group members' names and PIDs should be on each page of the submission.
  - Include the text of each question you are answering, followed by your solution.

## Topics:

1. Push-down Automata
2. Closure Properties of Context-Free Languages
3. Introduction to Turing Machines - Construction
4. Turing Machine Traceability & High-Level Design

**Problem 1 - Ratio Language:**

Construct a PDA for:

$$L = \{ 0^{2m} 1^{3m} \mid m \geq 0 \}$$

Explain how your PDA ensures that every pair of 0's corresponds to exactly three 1's.

**Problem 2 - Mirror Around a Marker:**

Design a PDA that recognizes:

$$L = \{ w\#w^R \mid w \in \{0, 1\}^* \}$$

Provide either a formal  $\delta$ -definition or a clearly labeled diagram.

**Problem 3 - Interleaved Dependence:**

Construct a PDA that accepts:

$$L = \{ a^m b^n c^n d^m \mid m, n \geq 0 \}$$

**Problem 4 - Even-Length Centered Language:**

Give a PDA for:

$$L = \{ a^n b^{2n} a^n \mid n \geq 0 \}$$

**Problem 5 - Split-Sum with Noise:**

Construct a PDA (state diagram or formal 7-tuple) for the language:

$$L = \{ d^* a^i d^* b^j d^* c^k d^* \mid i = j + k, i, j, k \geq 0 \}$$

**Problem 6 - Closure under SUFFIX:**

Let  $A \subseteq \Sigma^*$  be any context-free language. Define:

$$SUFFIX(A) = \{ v \in \Sigma^* \mid \exists u \in \Sigma^* : uv \in A \}$$

Prove that the class of context-free languages is **closed under SUFFIX**.

**Problem 7 - Closure under Kleene Star:**

Let  $A \subseteq \Sigma^*$  be any context-free language. Prove that  $A^*$  is context-free by giving a generic construction that, from a PDA (or CFG) for  $A$ , builds a PDA (or CFG) for  $A^*$ .

Your proof must:

1. describe the construction precisely, and
2. justify correctness in both directions ( $\subseteq$  and  $\supseteq$ ).

**Problem 8 - Closure under Rotational (Cyclic) Closure:**

For any language  $A \subseteq \Sigma^*$ , define its **rotational closure**:

$$RC(A) = \{ yx \mid x, y \in \Sigma^*, xy \in A \}$$

Show that if  $A$  is context-free, then  $RC(A)$  is also context-free.

**Problem 9 - Closure under Reversal:**

Let  $A$  be **context-free** language generated by  $G = (V, \Sigma, R, S)$ . The **Reversal** of  $A$  is defined as:

$$A^R = \{ w^R \mid w \in A \}, \text{ where } w^R \text{ denotes the reverse of string } w.$$

Prove that the class of context-free languages is closed under Reversal.

## Problem 10 - Non-closure under Intersection and Complement:

Prove, in general, that CFLs are not closed under **intersection** and hence not closed under **complement**.

- First, exhibit CFLs  $L_1, L_2$  such that  $L_1 \cap L_2$  is not context-free (justify via a standard non-CFL and intersection with a regular language if useful).
- Then use **De Morgan's laws** together with the known closure of CFLs under **union** to argue that closure under complement would imply closure under intersection, yielding a contradiction.
- You can use the following fact without proof:  $L = \{a^n b^n c^n \mid n \geq 0\}$  is **not** context-free.

**In this section:** Give an implementation-level description of the TM or draw the state diagram

### Problem 11 - Triple-Match Language:

Design a Turing Machine that decides:

$$L = \{ a^n b^n c^n \mid n \geq 0 \}$$

Describe how the TM repeatedly crosses off one a, b, and c in each cycle, rejecting any mismatch and halting when all symbols are marked.

Explain why this language is **not CFL** but **Turing-decidable**.

### Problem 12 - Balanced Binary Strings:

Construct a TM that decides:

$$L = \{ w \in \{0, 1\}^* \mid \#0 = \#1 \}$$

Give a formal description showing how the machine locates one 0 and one 1 at a time, marking them off until none remain.

Argue briefly why the TM always halts.

### Problem 13 - Equality Across Marker:

Design a TM that decides:

$$L = \{ w\#w \mid w \in \{0, 1\}^* \}$$

Describe how your TM scans left and right across the # symbol, comparing corresponding characters.

### Problem 14 - Copy Language:

Design a Turing Machine that decides:

$$L = \{ ww \mid w \in \{0, 1\}^* \}$$

### Problem 15 - Unary Multiples:

Design a Turing Machine that decides:

$$L = \{ 0^{3n} \mid n \geq 0 \}$$

### Problem 16 - Configuration Tracing:

Given the transition function  $\delta$  below, trace the complete sequence of configurations when the input is **001**.

$$\begin{aligned}\delta(q_0, 0) &= (q_1, X, R), \\ \delta(q_1, 0) &= (q_1, 0, R), \\ \delta(q_1, 1) &= (q_2, 1, L), \\ \delta(q_2, X) &= (q_{\text{accept}}, X, R)\end{aligned}$$

List every configuration until the machine halts, showing the current state, tape contents, and head position.

### Problem 17 - Binary Addition:

Provide an implementation-level description for a TM that decides:

$$L = \{ a\#b\#c \mid a + b = c, a, b, c \text{ in binary} \},$$

where leading 0's are allowed.

Explain how your TM simulates binary addition with carry bits while scanning from right to left, and argue that it halts on all inputs.

### Problem 18 - String Reversal:

Describe a Turing Machine that, on input  $w \in \{0, 1\}^*$ , halts with  $w^R$  written on the tape.

Explain using implementation-level description how the TM can repeatedly move the last unprocessed symbol to the left side until the string is reversed, and why this process terminates.

### Problem 19 - Palindrome Checker:

Design a TM that decides whether a binary string is a palindrome.

Provide an implementation-level description explaining how the TM marks matching outer symbols and moves inward.

Briefly justify that the machine always halts.

## Problem 20 - Equal Counts Across Alphabet:

Consider:

$$L = \{ w \in \{0, 1, 2\}^* \mid \#0 = \#1 = \#2 \}$$

Explain why this language cannot be recognized by a single-stack PDA. Then outline using implementation-level description how a two-stack or Turing-machine-style model could succeed by tracking two counts at a time.