Modern Theory of Computation (22IST62) [AY:2024-25]
CCE 2
Assignment Date: 28/05/2025
Submission Date: 9/06/2025

Practical Assignment Questions Using Python

Submission Guidelines

- All answers must be **typed and submitted in LaTeX format**.
- You may use **Overleaf** (https://www.overleaf.com) or any LaTeX editor of your choice.
- The final submission should include:
  - ✓ Clearly labeled sections for each question.
  - ✓ Proper use of mathematical notation where applicable.
  - ✓ The .tex file and compiled .pdf should both be submitted.

Note: ***Assignments not submitted in LaTeX will be considered incomplete.***

Question 1: Simulating a DFA [L3][CO1,2][5 Marks]

Objective: Implement a **Deterministic Finite Automaton (DFA)** to recognize whether a given binary string is divisible by 5.

$$\boldsymbol{L = \{w \in \{0,1\} *| \ the\ decimal\ value\ of\ w\ is\ divisible\ by\ 5\}}$$

Instructions:

- Define a **DFA as a transition table** in Python.
- Implement a function that **takes a binary string as input** and checks whether it is accepted or rejected by the DFA.
- Example Input: " 10000", Expected Output: "Rejected"
- Example Input: "1010", Expected Output: "Accepted"

Question 2: Context-Free Grammar Parser (CFG). [L4][CO2] [5 Marks]

Write a Python program that:

- Accepts a CFG in the form of production rules for the given language

$$\boldsymbol{L = \{WW^R\}\ U\ \{W(a+b)W^R\}\ W \in \Sigma^*, where\ \Sigma = \{a, b\}}$$

- Takes an input string and Checks whether the string belongs to the language defined by the grammar using leftmost derivation.
- Test Case: $aabbbbaa$, abababa

Question 3: Pushdown Automata (PDA) Simulation [L3][CO3] [5 Marks]

Write a Python program that simulates a PDA to recognize the language

$$\boldsymbol{L = \{ a^{n} b^{m} c^{m} d^{n} \mid n \geq 1, m \geq 1 \}}$$

Test Case:

Input: $\boldsymbol{aabbccdd, aabcdd, aaabbccddd, etc.}$

Question 4: Turing Machine Simulator [L5][CO4,5] [10 Marks]

Objective: Implement a **Turing Machine simulator** that recognizes whether a binary string has **equal numbers of 0s and 1s**. Additionally, mention a real-world application where this kind of Turing Machine logic (e.g., balancing counts, tracking pairs) can be useful, such as in compilers, verifiers, or DNA pattern analysis **Instructions:**

- The machine should scan the tape, **replace 0s and 1s** to keep track of counts, and decide **accept/reject**.
- Input Example: "1010" → **Accepted**
- Input Example: "110" → **Rejected**