Turing Machine Simulator – Project Report

Team Members' Names and IDs

- **Yahia Hany Gaber – 231000412**
- **Ahmed AbdelMaboud – 231000224**
- **Abdelrahman Yasser – 231000102**
- **Mazen Ahmed – 231001075**

Detailed Description of the Project

This project implements a **Turing Machine Simulator** capable of reading a Turing Machine definition, parsing it, constructing its components, and simulating its full execution step-by-step. The simulator allows users to test whether an input string belongs to a given language by observing the full transition sequence, including tape updates and head movements.

The simulator takes a Turing Machine description from a text file or input, parses it into states, alphabet, transitions, and final states, then runs the simulation on a user-provided input string. It outputs the sequence of configurations (tape content, head position, and current state) until the machine reaches an accepting or rejecting state. The project demonstrates how theoretical models such as Turing Machines can be implemented programmatically.

Input Format

1. **TM Definition File**
   A text file containing:
   - Machine name
   - Start state
   - Final/accept states

   Transition rules in the format:
   - current_state, read_symbol: next_state, write_symbol, direction
2. **Input String**
   A string (e.g., 0011) representing the tape's initial content.

Output Format

- A list of all simulation steps including:
  - Current head position.
  - Current tape (list of characters).
  - Current state.
  - Next state.
- Result: **ACCEPT** or **REJECT**.

Inside Mechanism

The simulator is built from three main components:

1. **Tape Implementation**
    - The tape is initialized as a list with a blank _ on both ends.
      Example input 0011 becomes:
      ["_", "0", "0", "1", "1", "_"].
    - Methods implemented:
        - Move head Left/Right
        - Read current symbol
        - Write symbol
        - Return current tape state + head position
2. **Parser Module**
    - Takes a TM definition text file and converts it into a Python dictionary.
    - Extracts:
        - States
        - Alphabet
        - Tape alphabet
        - Transitions
        - Start state
        - Final states
3. **Simulation Engine**
    - Reads the parsed TM definition.
    - Applies transitions sequentially to the tape.
    - Records each step in a list of transitions, for example:
    - [head_position, tape_characters, current_state, next_state]
    - Stops when:
        - A final state is reached → ACCEPT
        - No valid transition exists → REJECT

Programming Language, Tools & Libraries Used

- **Programming Language:** Python
- **Libraries:**
    - No external libraries are used; project relies on pure Python data structures and logic.

Images of the Project with Output:

Figure 1- Simulator interface:

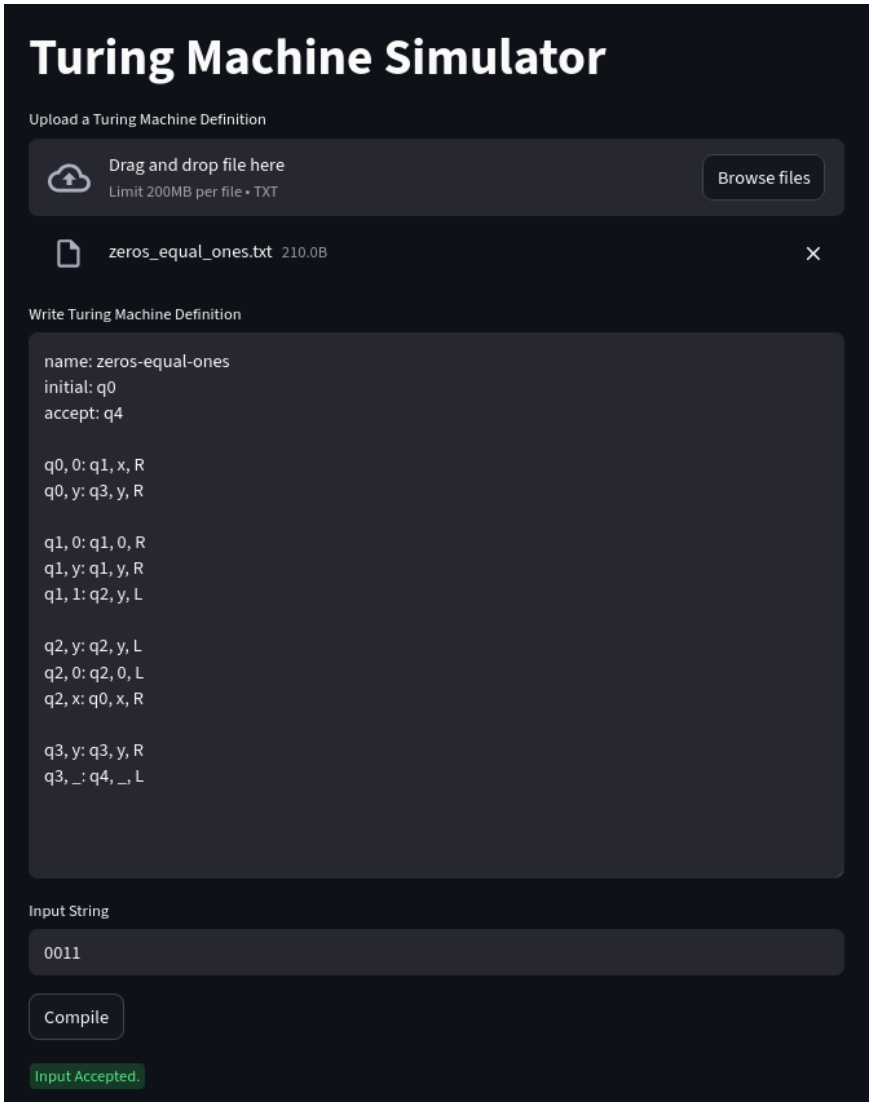

Figure2- Example: Input String = 0011:

Machine Definition

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$$

- Q = {'q4', 'q3', 'q1', 'q0', 'q2'}
- Γ = {'y', '0', 'x', '_', '1'}
- q_0 = q0
- F = {'q4'}
- δ =

Current State	Current Symbol	Next State	Written Symbol	Direction
q0	0	q1	x	R
q0	y	q3	y	R
q1	0	q1	0	R
q1	y	q1	y	R
q1	1	q2	y	L
q2	y	q2	y	L
q2	0	q2	0	L
q2	x	q0	x	R
q3	y	q3	y	R
q3	_	q4	_	L

Tape Transitions

Current State	Next State	Head Position	0	1	2	3	4	5
q0	q1	1	_	>0	0	1	1	_

Current State	Next State	Head Position	0	1	2	3	4	5
q1	q1	2	_	x	>0	1	1	_

Current State	Next State	Head Position	0	1	2	3	4	5
q1	q2	3	_	x	0	>1	1	_