Extra Credit Assignment Readme

Version 1 12/16/24

<table>
<tr><td>1</td><td>Team Name: N/A</td></tr>
<tr><td>2</td><td>Team members names and netids: kwilli33</td></tr>
<tr><td>3</td><td>Overall project attempted, with sub-projects: Determine whether a machine is decidable, recognizable, or co-turing recognizable—as well as finding its complexity.</td></tr>
<tr><td>4</td><td>Overall success of the project: Fairly successful</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete: 10hrs</td></tr>
<tr><td>6</td><td>Link to github repository: https://github.com/kw-nd/toc-ExtraCredit</td></tr>
<tr><td>7</td><td>List of included files (if you have many files of a certain type, such as test files of different sizes, list just the folder): (Add more rows as necessary). Add more rows as necessary.
<table>
<tr><td>File/folder Name</td><td>File Contents and Use</td></tr>
<tr><td colspan="2">Code Files</td></tr>
<tr><td>program.py</td><td>The code defines an AutomatonAnalyzer class that processes a CSV representation of a finite automaton to evaluate its properties—such as decidability, recognizability, and time complexity—and outputs the results.</td></tr>
<tr><td colspan="2">Test Files</td></tr>
<tr><td>test_program.py
machine.csv</td><td>test_program.py:
The code defines a testing framework for the AutomatonAnalyzer class, creating temporary CSV files representing various automata, running</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>analysis tests with expected results, and cleaning up the files afterward.

machine.csv:
The code defines a finite automaton in CSV format with states, transitions, and properties for start and accept states, representing a machine that processes binary inputs.</td></tr>
<tr><td colspan="2">Output Files</td></tr>
<tr><td>output.png
test_output.png</td><td>output.png:
Output of program.py

test_output.png:
Output of test_program.py</td></tr>
<tr><td colspan="2">Plots (as needed)</td></tr>
<tr><td>N/A</td><td>N/A</td></tr>
</table>
</td></tr>
<tr><td>8</td><td>Programming languages used, and associated libraries:
Language: Python

Libraries: csv, sys, sympy, os, subprocess</td></tr>
<tr><td>9</td><td>Key data structures (for each sub-project):
program.py: list, dictionary, set, tuple
test_program.py: list, dictionary</td></tr>
<tr><td>10</td><td>General operation of code (for each subproject)
program.py:
The program reads an automaton's state transitions from a CSV file, analyzes its properties (decidability, recognizability, co-Turing recognizability), calculates its time complexity, and outputs the results.

test_program.py:
The program creates test CSV files with predefined automaton configurations, runs program.py on each file, and validates the correctness of its output against expected results.</td></tr>
</table>

11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. Test Cases (test_program.py): **Test 1**: A single-state automaton with all transitions, validating basic functionality and recognizing fully connected states. **Test 2**: A two-state complete automaton ensuring proper handling of transitions and acceptance conditions. **Test 3**: An automaton with a missing transition, testing whether the program correctly identifies non-decidability. **Test 4**: A multi-state automaton triggering quadratic time complexity, verifying accurate complexity calculation. **Test 5**: A complex automaton with exponential paths, testing the program's ability to identify exponential time complexity
12	How you managed the code development **program.py**: organized into a class-based structure AutomatonAnalyzer, encapsulating functionalities like CSV loading, property checks, and complexity calculations, with a clean analyze method for execution. **test_program.py**: the code follows a modular approach with separate functions for creating CSV files, running the analyzer, and validating outputs, ensuring clear separation of test setup, execution, and verification.
13	Detailed discussion of results: **program.py:** The input file, machine.csv, contains a state machine defined in CSV format The automaton is not decidable, indicating at least one state lacks a transition for some input. It is recognizable, meaning there is a path to an accepting state. It is co-Turing recognizable, as the program assumes this property always holds. The time complexity is $O(n^2)$, showing the automaton's state transitions scale quadratically. **test_program.py:** Test 1 and Test 2 PASSED, confirming the program works for fully connected automata with single and two states. Test 3 PASSED, validating the program correctly detects missing transitions as making the automaton undecidable. Test 4 and Test 5 FAILED, indicating the program did not correctly identify the expected quadratic ($O(n^2)$) and exponential ($O(2^n)$) complexities.
14	How team was organized: N/A
15	What you might do differently if you did the project again: Start the project earlier so I would have more time to modify my code and pass all the test cases
16	Any additional material: N/A