Project Readme Template

Version 1 9/11/24

A single copy of this template should be filled out and submitted with each project submission, regardless of the number of students on the team. It should have the name readme_"teamname"

Also change the title of this template to "Project x Readme Team xxx"

<table>
<tr><td>1</td><td>Team Name: Kwilli</td></tr>
<tr><td>2</td><td>Team members names and netids: Kwilli</td></tr>
<tr><td>3</td><td>Overall project attempted, with sub-projects: Given a NTM and a string, trace all possible paths the NTM might have taken, and stop when either one of the possible paths ends at an accept state, or ALL possible paths end at a reject state.</td></tr>
<tr><td>4</td><td>Overall success of the project: Reasonably successful</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete: 16 hrs</td></tr>
<tr><td>6</td><td>Link to github repository: https://github.com/kw-nd/toc-project02</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>traceNTM_kwilli.py</td><td>Simulates the behavior of a Non-Deterministic Turing Machine (NTM) based on a configuration file, tracing all possible computation paths until it accepts, rejects, or reaches a specified step limit.</td></tr>
<tr><td colspan="2">Test Files</td></tr>
<tr><td>test_traceNTM_kwilli.py, machine_kwilli.csv</td><td>test_traceNTM_kwilli.py: Tests the traceNTM.py implementation by running predefined test cases with various input</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>strings and depth limits, validating the results (accept, reject, or stopped) and reporting the total transitions alongside a description of each test case.

machine_kwilli.csv: Defines a non-deterministic Turing Machine (NTM) called "a plus" that accepts strings consisting of one or more a’s, with states q1 (start), q2 (transition), and q3 (accept), and transitions that loop in q1 on a, move to q2 on a, and move to q3 on _ (blank) to accept.</td></tr>
<tr><td colspan="2">Output Files</td></tr>
<tr><td>test-image_kwilli.png</td><td>Output of test_traceNTM_kwilli.py</td></tr>
<tr><td colspan="2">Plots (as needed)</td></tr>
<tr><td>main-image_kwilli.png</td><td>Output of traceNTM_kwilli.py (table)</td></tr>
</table>
</td></tr>
<tr><td>8</td><td>Programming languages used, and associated libraries:
- Language: Python
- Libraries: csv, collections, subprocess</td></tr>
<tr><td>9</td><td>Key data structures (for each sub-project):
- dictionaries, lists, sets, strings (traceNTM_kwilli.py)
- lists, dictionaries, strings (test_traceNTM_kwilli.py)</td></tr>
<tr><td>10</td><td>General operation of code (for each subproject):
- traceNTM_kwilli.py: This program simulates a Non-Deterministic Turing Machine (NTM) by reading a machine description and input string, exploring all possible computational paths up to a given depth, and printing the result (accept, reject, or stopped) along with the computational path and transition details.
- test_traceNTM_kwilli.py: This program runs a series of predefined test cases by executing the `traceNTM.py` program with different inputs and depth limits, collects the results (accept, reject, or stopped), and summarizes the outcomes</td></tr>
</table>

	with the number of transitions and the reason for each test.
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. The test cases cover scenarios to validate the NTM's behavior, including accepting valid inputs, rejecting invalid or empty inputs, stopping execution due to depth limits, and handling single or large input strings within or beyond specified depth constraints.
12	How you managed the code development The main program (traceNTM_kwilli.py) implements the functionality of a Non-Deterministic Turing Machine, handling machine configuration, simulation, and output generation. The test program (test_traceNTM_kwilli.py) runs multiple predefined scenarios, using subprocesses to validate the main program's behavior against expected results, and summarizes outcomes for accuracy and performance. Together, they ensure the main program functions correctly while maintaining modularity and streamlined testing.
13	Detailed discussion of results: **main-image_kwili.png:** The output in the image shows the execution of (traceNTM_kwilli.py) where the string aaa is accepted in 4 steps, detailing the machine's configuration at each step (left of head, state, head character, right of head), the result (accept), and the total transitions (7). **test-image_kwilli.png:** The output in the image demonstrates the test program (test_traceNTM_kwilli.py) executing various predefined test cases on the traceNTM.py program, displaying the input string, result (accept, reject, or stopped), total transitions taken, and the reason for each test case to validate the behavior of the Turing Machine implementation.
14	How team was organized There was no team involved. I worked independently, handling all the programming, data management, and development tasks myself.
15	What you might do differently if you did the project again If I did the project again, I would start earlier, make the code more concise, and enhance it to handle multiple machines simultaneously.
16	Any additional material: N/A