Project 02 README Team mchou3

Version 1 9/11/24

1	Team Name: **mchou3**
2	Team members names and netids: **Matthew Chou - mchou3**
3	Overall project attempted, with sub-projects: **k-tape Turing Machine**
4	Overall success of the project: **Completed - Simulator Works**
5	Approximately total time (in hours) to complete: **15 hours**
6	Link to github repository: **https://github.com/matthewvchou/ktape_mchou3**
7	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.

File/folder Name	File Contents and Use
Code Files	
/code/ktapesim_mchou3.py	**Script that simulates a k-tape Turing Machine.**
Test Files	
/data/test_files/wcw_pass1.csv **/data/test_files/wcw_pass2.csv** **/data/test_files/wcw_pass3.csv**	**CSV files that hold test cases that will pass the simulation of the k-tape Turing Machine of {wcw | w ∈ {a,b}*} (i.e. the string input will be accepted by the machine)** **Each file holds a different test case**
/data/test_files/wcw_fail1.csv **/data/test_files/wcw_fail2.csv** **/data/test_files/wcw_fail3.csv**	**CSV files that hold test cases that will fail the simulation of the k-tape Turing Machine of {wcw | w ∈ {a,b}*} (i.e. the string input will be rejected by the machine)** **Each file holds a different test case**
/data/test_files/abc_pass1.csv **/data/test_files/abc_pass2.csv** **/data/test_files/abc_pass3.csv**	**CSV files that hold test cases that will pass the simulation of the k-tape Turing Machine of {a^nb^nc^n, n ≥ 0} (i.e. the string input will be accepted by the machine)**

		Each file holds a different test case
	/data/test_files/abc_fail1.csv **/data/test_files/abc_fail2.csv** **/data/test_files/abc_fail3.csv**	**CSV files that hold test cases that will fail the simulation of the k-tape Turing Machine of {a^nb^nc^n, n ≥ 0} (i.e. the string input will be accepted by the machine)** **Each file holds a different test case**
	Output Files	
	/data/output_files/wcw_pass1.csv **/data/output_files/wcw_pass2.csv** **/data/output_files/wcw_pass3.csv**	**CSV files that hold output of the test cases that will pass the simulation of the k-tape Turing Machine of {wcw | w ∈ {a,b}*} (i.e. the string input will be accepted by the machine)** **Each file holds the output of the file within the test_files directory of the same name**
	/data/output_files/wcw_fail1.csv **/data/output_files/wcw_fail2.csv** **/data/output_files/wcw_fail3.csv**	**CSV files that hold output of the test cases that will fail the simulation of the k-tape Turing Machine of {wcw | w ∈ {a,b}*} (i.e. the string input will be accepted by the machine)** **Each file holds the output of the file within the test_files directory of the same name**
	/data/output_files/abc_pass1.csv **/data/output_files/abc_pass2.csv** **/data/output_files/abc_pass3.csv**	**CSV files that hold output of the test cases that will pass the simulation of the k-tape Turing Machine of {a^nb^nc^n, n ≥ 0} (i.e. the string input will be accepted by the machine)** **Each file holds the output of the file within the test_files directory of the same name**
	/data/output_files/abc_fail1.csv **/data/output_files/abc_fail2.csv** **/data/output_files/abc_fail3.csv**	**CSV files that hold output of the test cases that will fail the simulation of the k-tape Turing Machine of {a^nb^nc^n, n ≥ 0} (i.e. the string input will be accepted by the machine)** **Each file holds the output of the file within the test_files directory of the same name**
8	Programming languages used, and associated libraries: **Language: Python**	

	Libraries: sys, csv
9	Key data structures (for each sub-project): **Dictionaries, Tuples, Lists, Sets, Strings, Generators**
10	General operation of code (for each subproject) **ktapesim_mchou3.py simulates the operation of a k-tape Turing machine using transition rules specified in a CSV file. The script initializes the machine with metadata such as the number of tapes, input string, states, start state, and accepting states, then dynamically processes the input based on defined transitions. It reads and writes to multiple tapes, moves tape heads according to transition rules, and supports wildcards within transitions as well. The simulation outputs a step-by-step trace of the machine's states, tape contents, and head movements, concluding with whether the input string is accepted or rejected.**
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. **For this project, I tested using k-tape Turing machines for the languages {wcw | w ∈ {a,b}*} and {a^n b^n c^n, n ≥ 0}, using a total of 12 test cases - 6 for each machine. Each set included 3 passing and 3 failing test cases. For both languages, I designed 2 edge cases and 1 conventional case within the passing and failing categories to test the machines' output.** **The passing test cases were made to ensure the machines could accept strings that conformed to the language. These included smaller inputs for edge cases (e.g., the smallest valid strings like "c" and "aca" for the wcw machine) and a conventional example with a larger, structured input that required more transitions (e.g., "aaabbbcaaabbb" for the wcw machine). These cases tested the machines' ability to handle various valid patterns.** **The failing test cases were used to ensure the machines rejected strings that violated the language. Edge cases included strings with subtle errors, such as "aabc" and "abcabc" (missing structure or extra characters), while the conventional failing cases tested larger, clearly invalid inputs like "cba". These failing cases helped make sure that the machines correctly halted and rejected invalid inputs.** **By including both passing and failing test cases with different levels of complexity, I was able to thoroughly check how well the k-tape Turing machines performed. The edge cases helped test tricky boundary scenarios, while the more typical cases showed that the machines could handle standard examples as expected. Overall, this approach made it easier to see how reliable and robust the machines were.**
12	How you managed the code development **First developed the initialization of the k-tape machine. Then, I moved on to implementing the simulator itself (i.e. allowing the machine to iteratively step**

	through the input). Finally, I created the script's ability to print out each step and also print its information.
13	Detailed discussion of results: **For each test case, the output was as follows:** **Machine Name: {Machine Name}** **Input String: {Input String}** **Start State : {Start State}** **------------------** **Step 1** **Tape 0: ...** **...** **Tape n: ...** **------------------** **Step 2** **...** **And so on.** **To indicate where the tape head was for each tape, I used a '^' that would point to the respective position on that tape underneath the tape itself.** **For the failing test cases, I made sure to stop the script as soon as an invalid transition was encountered. This approach allowed me to show exactly where the machine would fail and ensured that the script didn't keep running indefinitely on invalid inputs. I simulated the entire machine until it encountered the blank character ('_') on the first tape, which marked the end of the input.**
14	How team was organized **No team, did by myself**
15	What you might do differently if you did the project again **If I were to do the project again, I would probably allocate more time into developing more test cases. It would be interesting to see the full capabilities of what I made with more complex test cases.**

16	Any additional material: **NA**