Project 02 Readme Team Slama

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_slama
Also change the title of this template to “Project x Readme Team xxx”

1	Team Name: **slama**
2	Team members names and netids: **Sophie Lama, slama**
3	Overall project attempted, with sub-projects: **Program 1: Tracing NTM Behavior**
4	Overall success of the project: **This project seems to have been a success. For the two test files I was working with, the main program correctly traces how each file’s nondeterministic Turing machine would behave given some input string.**
5	Approximately total time (in hours) to complete: **6-8 hours**
6	Link to GitHub repository: **https://github.com/siscalie/theory_project02**
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	
traceNTM_slama.py	**This file contains the main program that traces the behavior of a nondeterministic Turing machine.**
Test Files	
check_a_plus_slama.csv **check_abc_star_slama.csv**	**These two CSV files represent two different Turing machines, one for an a^+ language and one for an a*b*c* language.**
Output Files	
output_a_plus_slama.txt **output_abc_star_slama.txt**	**These two text files contain the output of various instances the main program was run. There is one output file for various input strings with the a^+ language, and one file for strings with the a*b*c* language.**

	Plots (as needed)
8	Programming languages used, and associated libraries: **Python, with ‘sys’ used to access stdin as a stream of input.**
9	Key data structures (for each sub-project): **The most important structure for the project was a Machine object I created, which represents the Turing machine that is represented by a CSV file the user enters. Each Machine has a name string, a start state, an accept state, a reject state, and a list of possible transitions.**
10	General operation of code (for each subproject): **Running the main program begins with asking the user for some input. The user can enter the name of the CSV file of the Turing machine they want to run, as well as an input string to run on the machine. They also enter in the limit, which is the maximum depth for the tree of possible configurations.** **Next, the CSV file is read, and the attributes of the machine are added to a Machine object and returned to the main function. After printing some initial information about the machine and string, we begin to iterate through the tree of possible configurations, trying to find a path that leads to an accept state.** **This iterate function is recursive, and it returns the final state of the best possible path, as well as the path’s depth in the tree of configurations and the sequence of configurations for the path. If this was an accept path, then the list of configurations will be printed, as well as the depth of the accept path. If this was a path that exceeded the depth limit, this information will be printed to the user. Finally, if no accept path was found and none exceeded the limit, then the longest reject path’s depth will be printed out to the user.**

11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. **The first test case I used was the a_plus CSV file provided by another student. This simple Turing machine helped me create the basic structure and organization of my program, in particular the recursive iterate function. This machine helped me a lot in knowing when I hit an accept state and when to return from the function. Once my program seemed to work well with the a_plus machine, I tested it with the abc_star CSV file (which was also provided by another student and shared in a Canvas announcement). I was able to fix little quirks in the program using this more complicated Turing machine, such as creating an accurate list of configurations for the path ultimately taken by the machine.**
12	How you managed the code development: **This was a solo project, so I wrote the code individually. I began my work by creating functions to take in user input from stdin and to read in information from a CSV file. The bulk of my work was**

	spent creating the iterate function. The end of the work involved formatting the results of the iterate function.
13	Detailed discussion of results: **SEE DISCUSSION DOCUMENT detailed_discussion_slama.pdf.**
14	How team was organized: **Solo project**
15	What you might do differently if you did the project again: **I would probably try to rewrite the code to be more efficient. The way it is now, it gets the job done, but I’m sure there are better, faster ways to get the job done.**
16	Any additional material: