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"

1	Team Name: iansntm
2	Team members names and netids: Ian Robinett, irobinet
3	Overall project attempted, with sub-projects: Program 1: Tracing NTM Behavior
4	Overall success of the project: The project was successful. I completed the project and passed several tests with different input files. The tracing functionality for a Non-deterministic Turing Machine was implemented and validated.
5	Approximately total time (in hours) to complete: 8
6	Link to github repository: https://github.com/irobinett3/traceTM_iansntm
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	
main_iansntm.py	This contains my main algorithm. It contains a function that read in NTM files in csv form and stores them. It also contains a function that sees if an NTM will accept a given string, performing a BFS and showing the configurations at each depth.
Test Files	
testing_file_iansntm.py	This is a file containing a variety of tests using the pytest library.
all .csv files in csv_tests_iansntm/	This is a directory containing the NTM files provided by Professor Kogge
Output Files	

	<table><tr><td>output_iansntm.txt</td><td>This contains the output from an example I ran the machine on. It contains information about the input string and the machine, as well as determining whether or not the machine accepts the string, and showing each state of the BFS.</td></tr><tr><td colspan="2">Plot (as needed)</td></tr><tr><td>N/A</td><td>My project did not analyze time complexity, so there was no need for this</td></tr></table>
8	Programming languages used, and associated libraries: Languages: Python Libraries: unittest, csv, deque (from collections)
9	Key data structures (for each sub-project): Deque: Used to store configurations of the tape and Turing machine states during the execution. List: Used for the tape representation and states.
10	General operation of code (for each subproject) The program reads a Turing machine description from a CSV file using the read_tm_csv function. It gets the name of the machine, its set of states, input alphabet, tape alphabet, start state, accept state, and reject state. It then simulates the behavior of a Non-deterministic Turing Machine (NTM) on an input string using the trace_ntm function. It records the configurations at each stage in the BFS. It outputs the acceptance status of the input string, the number of transitions taken, and the configuration tree.

11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. Test Cases: I added a variety of test cases for the different NTM csv files that were provided for us. For each NTM, I provided input strings that were both false and true. This allowed me to check behavior in both cases. I also checked edge cases, including empty strings, enabling me to test the NTM's behavior in all cases Empty String ("_"): To check if the machine handles an empty input. These test cases validated that the machine properly handles both valid and invalid strings, and correctly detects acceptance/rejection.

12	How you managed the code development: The development was done incrementally: First, implemented the function to read the machine description from the csv files. Then, traced the machine's behavior on an input string. Finally, I validated the behavior using multiple test cases and created a pytest file to automate this process.
13	Detailed discussion of results: The machine correctly simulated the NTM's behavior, tracing possible configurations at each depth. It also correctly identified acceptance and rejection of input strings. The results were consistent with the expected behavior based on the machine description in the CSV files.
14	How team was organized I worked entirely alone and completed the project by myself.
15	What you might do differently if you did the project again Overall, I really enjoyed how my project went. If I were to change one thing, however, i would have randomly generated some of my own problems, but the provided examples given to me sufficiently proved to me that my code was sufficient.
16	Any additional material: The complete code, test cases, and results can be found in the linked GitHub repository.