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: izmo
2	Team members names and netids: Izzy Molnar – imolnar
3	Overall project attempted, with sub-projects: NTM Trace
4	Overall success of the project: Project was successful as results produced from the program matched those from my hand calculations.
5	Approximately total time (in hours) to complete: 5 hours
6	Link to github repository: https://github.com/izzymolnar/Project2-TOC
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	
src/ntm_tracer.py	Main program, contains primary functionality of program
Test Files	
input/aplus.csv	NTM that accepts strings with 1 or more a's
input/composite.csv	NTM that tests whether a number represented by a string of 1's is composite or prime
input/equal_01s.csv	NTM that accepts a string with the same number of 1's and 0's
Output Files	
output/aplus.txt	Text file with output from aplus.csv test script
output/composite.txt	Text file with output from composite.csv test script
output/equal_01s.txt	Text file with output from equal_01s.csv test script

output/aplus.jpeg	Screenshot of output from aplus.csv test script
output/composite.jpeg	Screenshot of output from composite.csv test script
output/equal_01s.jpeg	Screenshot of output from equal_01s.csv test script

8	Programming languages used, and associated libraries: Python3, no additional libraries
9	Key data structures (for each sub-project): tree – List of lists to represent the BFS tree, config/initial_config – List representing the TM configuration, current_level and next_level – Lists of configurations path – List used in print_trace_path to store backtracked path branches – List of transition dictionaries
10	General operation of code (for each subproject): Simulates an NTM using BFS. Starts with an initial configuration and builds a tree where each level contains all possible configurations at that depth. Each iteration processes all configurations at the current level: if a configuration is in the accept state it backtracks to print the path and returns; if it is in the reject state it skips it; otherwise, it finds a valid transition from the transition table, generates the child configurations, and stores the parent information. If all the paths are rejected with no next level, the string is rejected.
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. I used the aplus.csv, composite.csv, and equal_01s.csv test cases that were provided to test a variety of different inputs. These test cases verified the accuracy of my program as their results matched the results from my hand calculations.
12	How you managed the code development: Since I was the sole developer on this project, I completed all the programming on the main branch and pushed my results when I was finished.
13	Detailed discussion of results: This program correctly simulates an NTM and finds accepting paths for various languages. For the aplus machine with input “aaaaa,” it accepts at depth 6. The program demonstrates that the machine stays in state q1 while reading a’s, it moved to q2 at the blank, and then q3 to accept. For the composite machine with “111111,” it accepts at depth 27, confirming 6 is composite by testing different factors. For the equal_01s machine with “110001,” it accepts at depth 27 and marks 0s and 1s with x’s until all are matched.
14	How team was organized: Individual project, so I completed all work.
15	What you might do differently if you did the project again: I would add some additional test cases beyond the three that I already have to test my program on a new set of inputs. I would also potentially prune rejected branches earlier to save memory, which would become more important with larger input files.

16	Any additional material: N/A