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: esotka
2	Team members names and netids: Ethan Sotka esotka
3	Overall project attempted, with sub-projects: Program 3 - TM RESET
4	Overall success of the project: Successful, working for test cases
5	Approximately total time (in hours) to complete: 7 hours
6	Link to github repository: https://github.com/esotka1/Project2-TOC.git
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	
tm_reset_esotka.py	Utilizes TuringMachineSimulator to create a Turing machine
main.py	Calls entrypoint.py
entrypoint.py	Parses arguments, calls tm_reset_esotka.py
turning_machine.py	Contains the class "TuringMachineSimulator", which includes code to emulate the features that make a Turing machine what it is (in the forms of functions called load_machine,

	get_transitions)
argument_input.py	Holds a function that has the code to read in arguments from the user
Test Files	
equal01.csv	Contains information representing a TM that can check for the Language L = {(0 U 1)* where there are the same amount of 0s as 1s}.
Output Files	
Plots (as needed)	

8	Programming languages used, and associated libraries: Python, No additional libraries for my code
9	Key data structures (for each sub-project): Lists
10	General operation of code (for each subproject): My code is based on the TuringMachineSimulator Class. Within my code, I start by initializing all the information my turning machine will need. I then enter a loop that limits the max steps. Inside the loop, I gather potential transitions from the state I am in with corresponding inputs. I then move to the next state based on this transition, or reject if there is no transition for my input. I then reset my head to 0 (rather than -1) if the TM attempts to move left. Finally, if I am in a reject or accept state, I break out of the loop and return my results.
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. I tested on the Language L = {(0 U 1)* where there is the same amount of 0s as 1s}. I knew that this language would result in many left movements from my TM, which would truly test how well it worked. In addition, there were many diverse test cases I could use that would ensure that my code was free of bugs. These tests proved that my code could handle a complex language despite being “limited” in having to reset on a left input

12	How you managed the code development In order to develop my code, I utilized in-code checks to see if I was receiving my desired results. For example, I would check the value for the head of the TM to ensure it was moving in the way that was expected. In addition to these in-code checks, I created my test file first so that I could utilize it in validating that different aspects of my code worked early on.
13	Detailed discussion of results: The turning machine did well for the test case I gave, and most likely was able to solve the language in fewer steps. This is due to not having to use several steps to return from its current place back to the beginning. In a situation where a TM often needs to reset to the beginning, we could see a use case for what I created.
14	How team was organized: I worked solo, so I instead had to focus on doing good work by myself. This consisted of good time management and planning.
15	What you might do differently if you did the project again: I believe I could have improved on knowing when to take a break and look at the problem again. I found that stepping away from a stubborn problem often yielded a solution much faster than simply powering through it.
16	Any additional material: