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: newdrop
2	Team members names and netids: Danny Mendler (dmendler)
3	Overall project attempted, with sub-projects: NTM tracing behavior
4	Overall success of the project: Much success
5	Approximately total time (in hours) to complete: 6 hours
6	Link to github repository: https://github.com/dmendler/new_drop
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	
traceTM_newdrop.py	Functions to read in a TM and simulate that TM on an input string
Test Files	
test_cases	Test cases for different Turing Machines
data_files	Data files for runs
Output Files	
output_files	Outputs files from traceTM_newdrop.py, using 2 different TMs and 4 input strings for each TM
Plots (as needed)	

<table>
<tr><td></td><td><table><tr><td>plots_newdrop.pdf</td><td>Table of statistics also found in output files</td></tr></table></td></tr>
<tr><td>8</td><td>Programming languages used, and associated libraries:
Python: csv, sys, collections</td></tr>
<tr><td>9</td><td>Key data structures (for each sub-project):
classes, lists, tables, defaultdict, deque</td></tr>
<tr><td>10</td><td>General operation of code (for each subproject)
First code reads in Turing Machine from .csv file. It then reads in an input string, as well as max depth and max transitions. It will then simulate that turing machine on the input string, exhausting every possible path. The program will either accept the machine (and print out necessary data), or reject the string.</td></tr>
<tr><td>11</td><td>What test cases you used/added, why you used them, what did they tell you about the correctness of your code.
I used the test files provided by the instructor. However, I only used 2 of the turing machine .csv files, while using multiple input strings per turing machine. I tested the validity of the turing machine .csv file by using different input strings, with some being accepted and some being rejected. This also checked the validity of the program by checking if the output created is correct (accept/rejected).</td></tr>
<tr><td>12</td><td>How you managed the code development:
I did all of the code myself, so I started working on the code on Friday, working a little bit on Saturday, and finished the table/README files on Sunday.</td></tr>
<tr><td>13</td><td>Detailed discussion of results:
The results print out the machine name, the initial string, how many transitions it took to accept the string, the tree depth, each configuration and the number of configurations, and the average non-determinism. The program shows that it correctly accepts strings that are accepted by the different turing machines and rejects strings that are rejected by those machines.</td></tr>
<tr><td>14</td><td>How team was organized: I was the only team member</td></tr>
<tr><td>15</td><td>What you might do differently if you did the project again:
I would change the code so the input could be sent in from a file, instead of having to manually input the file name, string, max depth, and max transitions every time. I would also change the formatting of the output, so it could be more clear to the reader.</td></tr>
<tr><td>16</td><td>Any additional material:</td></tr>
</table>