Project 2 Readme Team YouAreMine

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"

<table>
<tr><td>1</td><td>Team Name: YouAreMine</td></tr>
<tr><td>2</td><td>Team members names and netids: Clint Murray - cmurra26</td></tr>
<tr><td>3</td><td>Overall project attempted, with sub-projects: NTM Trace</td></tr>
<tr><td>4</td><td>Overall success of the project: Successful</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete: 8 hrs</td></tr>
<tr><td>6</td><td>Link to github repository: https://github.com/cmurra26/Project2-TOC</td></tr>
<tr><td>7</td><td>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.
<table>
<tr><td>File/folder Name</td><td>File Contents and Use</td></tr>
<tr><td colspan="2">Code Files</td></tr>
<tr><td>src</td><td>ntm_tracer_YouAreMine.py</td></tr>
<tr><td colspan="2">Test Files</td></tr>
<tr><td>input</td><td>aplus.csv
composite.csv
ntm.csv</td></tr>
<tr><td colspan="2">Output Files</td></tr>
<tr><td></td><td>output.txt</td></tr>
<tr><td colspan="2">Plots (as needed)</td></tr>
<tr><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>8</td><td>Programming languages used, and associated libraries: Python</td></tr>
<tr><td>9</td><td>Key data structures (for each sub-project):
NTM Trace</td></tr>
</table>

	Lists of four elements were used to hold the left side of the tape, the right side of the tape, the current head of the tape, and its parent index. The lists were then organized into a tree. Each level in the tree symbolized the BFS depth for a particular list. The Turing machine transitions were stored in a dictionary where the keys were the state and symbol and the values were lists of transition descriptions.
10	General operation of code (for each subproject) NTM Trace For the tracer, the BFS search algorithm is run over the NTM searching for an accepting state. The search begins from the first configuration and goes down the tree looking for any accepting or rejecting states. If the program can not find either, then the program creates new configurations with updated tape content. It ends its search when it finds an accepting state, all branches get rejected, or the maximum depth of the tree has been reached. The program displays its result.
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 composite.csv and the aplus.csv to validate the correctness of my program. I also verified the results by tracing through the machine myself. For aplus.csv, I tested the string ‘aaa’, and for composite.csv, I tested the string ‘111111’. The results that I expected matched with the output of the composite.csv and aplus.csv. Then, I created ntm.csv, where I checked if there were equal amounts of 0s and 1s. For this, I tested ‘00111”. I compared my prediction to the output of the program, and the results matched. I predicted that the string would not be accepted, and my program rejected it after 31 steps. This showed that my code was working very well.
12	How you managed the code development: Prior to coding in VSCode, I generated some pseudocode examples of what the print function and tracing functions might have to look like. The pseudocode was useful because I was able to easily implement them into actual code for the project.
13	Detailed discussion of results: Since my predictions matched with all the results from my code, I could safely say that my code was working properly. As can be seen in the output file, the number of steps and the number of transactions can be shown. When my program tested aplus.csv, ‘aaa’ was accepted in 4 steps. When my program tested composite.csv, “111111” was accepted in 31 steps. This makes sense because the composite problem was much more complex than the aplus problem because it had more states and more transitions to check. However, ntm.csv took just as long as the composite problem, but I also shortened the input string. The reason the ntm.csv problem takes the longest (if the inputs were of equal length) is because the machine must continuously check for unmarked symbols and mark them when they are used. This becomes time consuming for the program.
14	How team was organized: I did everything as I was the only team member.
15	What you might do differently if you did the project again: I would start earlier to possibly make the program more complex.
16	Any additional material: N/A