**EGID METALIA**

**ID:1320230**

**EXTRA CREDIT ASSIGNMENT**

**CSCI-312**

**Finite Automaton Interpreter Program Report**This program is designed to act like a finite state automaton. It is implemented with two input files-the transition table and the input string to be processed. The program will make its decision of whether or not to accept the input based on the rules stated within the transition table of said automaton as YES/NO, respectively. **Transition Table Handling**The transition table is loaded with the function load\_transition\_table,which reads the file line by line: the first line enumerates the alphabet-the allowed characters-and the rest of the lines enumerate how theautomaton changes its state depending on the allowed character. **Here's how that would look in code form:**1- Alphabet Extraction: split the first line into discrete characters to makeup the alphabet.  
2- State Transitions: Each line following the alphabet maps a state to its possible transitions. For instance, state 0 would move to state 1 when it reads b. All this information is stored in a nested dictionary where the outer dictionary keeps track of states and the inner one maps characters to their resulting states.  
3-Special States: The program picks up the start state-identifying it with S-and the final states-identifying them with F. These are stored separately so that the program knows where to start and what constitutes “accepting."

If the file is not in the correct format, such as missing transitions or invalid states, the function raises an error. This cleans up the input data before processing.  
  
**Input String Handling**The load\_input\_string function simply reads the second file, trims any extra spaces, and returns the string. It's straightforward but ensures no unexpected whitespace causes problems. **Processing the String  
  
Once the transition table and input string are loaded, the process\_string function takes over. Here's what it does:**1-Start at the Initial State: The automaton begins at the start state.  
2-Processing Character-by-Character: For each character in the string, based on the current state's transitions, it decides where to go next. If the character does not exist in the rules of the current state or there is no defined transition, it halts and returns No.  
3-Final Check: In case the string is fully processed and the automaton ends up in a final state, the string is accepted (Yes). Otherwise, it is rejected (No).

**Main Flow of the Program**The \_\_main\_\_ section brings it all together. It first loads the transition table and input string, then passes them to the processing function. The result (Yes or No) is printed at the end. Debugging tools are included but commented out in the final version for cleaner output.

**NOTE: While I was building this program, I faced errors, so during this time, I used the print function multiple times to check if the program was working the right way to read through the file. I did not remove those printing functions from the project and instead left them as comments.**