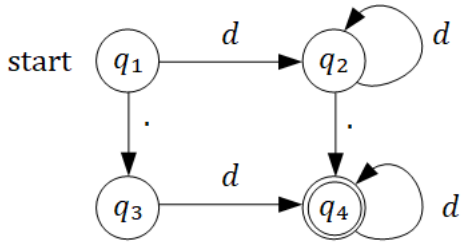
CS390 Principles of Programming Language

Assignment 2

Background:

Consider the following finite state automaton (Scott, 2016), which recognizes the language that consisting of digits with a single decimal point,



Formally, a ***Deterministic Finite Automaton*** (FDA) is as quadruple $\mathcal{A}$ over an alphabet Σ,

$$\mathcal{A} = \langle Q, q_s, F, \delta \rangle$$

where Q is a set of ***states***, $q_s \in Q$ is an ***initial*** state, F is a set of ***final*** states, and δ is a ***transition function*** from a current state and input symbol to a next state, $\delta: Q \times \Sigma \to Q$. For example, the automaton depicted above is defined as,

$$Q = \{q_1, q_2, q_3, q_4\}$$
$$q_s = q_1$$
$$F = \{q_4\}$$
$$\delta = \{\langle q_1, d, q_2\rangle, \langle q_1, ., q_3\rangle, \langle q_2, d, q_2\rangle, \langle q_2, ., q_4\rangle, \langle q_3, d, q_4\rangle, \langle q_4, d, q_4\rangle\}$$

Where $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$ and $d \in \Sigma$. Hence, in δ above, we should have defined,

$$\langle q_1, 0, q_2\rangle, \langle q_1, 1, q_2\rangle, \langle q_1, 2, q_2\rangle \ldots$$

Functional Requirements:

1. Create a C language program that can be used to construct any arbitrary Deterministic Finite Automaton corresponding to the FDA *definition* above.
   a. Create structs for the: *automaton*, a *state*, and a *transition*. For example, the automaton should have a "*states*" field, which captures its set of states as a linked list.

2. Prompt the user and ask whether the would like to create a new DFA, by entering its states, the initial state, its final states, and its transitions (one at a time) of load an existing DFA from a file. Prompt the user for a file name and save newly created DFAs to this file. For example, you should be able to create and save the DFA example above.

3. Prompt the user for an input strings, until they decide to quit, and use the DFA from Step 2 to determine whether the user's string is accepted by the DFA or not. Output this result. Recall, an input string is accepted, if beginning in the initial DFA state, the input transitions to a final state in the DFA, and no additional input is left.

4. Your program should be modular and minimally include functions for creating a new DFA, loading an existing DFA, and executing the DFA on the user's input string.

Non-Functional Requirements

1. Analyze each of the function in your program, including the main() function. For each function give the Big-Oh complexity function that captures the complexity of the DFA. What is the complexity class of this function (e.g. $O(1)$, $O(\log n)$, $O(n)$, $O(n^2)$, etc.?

2. Your program must demonstrate the use of C pointers, structs, malloc, sizeof, free printf, scanf, fscan, macro constants (e.g. #define), typedef. It must also demonstrate function calling and include a function prototype. Use an **array** to implement the "list" of states in the DFA, but use a **linked list** to implement the "list" of transitions in the DFA. You may assume the DFA will no more than 25 states.

3. Your program **must** compile and execute. Comment out and document for the faculty any problematic statements prior to submission.

4. Appropriately document your code using comments (minimally the program and each of the functions).

5. Implement this assignment in a single C file that executes using DevC++.

6. Your program **must** be your own individual work. Any external assistance (i.e. reading material on the Internet, receiving tutoring help from Regis tutors, etc., must be documented. Your faculty and the CS department reserve the right to reassess you, and, if necessary, re-determine your grade for this assignment, based on any additional assessments, e.g. via an oral exam).

This is strictly a C program (vs. C++). Additionally, do not use any predefined data structures when implementing the DFA (i.e.. create your own structs, linked lists, etc.)

Submission:
Submit your DevC++ source file <yourName_Assignment_2> to the Assignment 2 Dropbox in the Worldclass course shell associated with your current CS390 Section. (Although you will not earn points for testing, you should appropriately test your code for all requirements since you may not earn points, if it doesn't meet all the requirements.).

CS 390 Principles of Programming Languages

Assignment 2 Rubric

Assignment 2 DFA C program design, implementation, and analysis

Assignment	Exemplary	Advanced	Proficient	Not Demonstrated or Major Issues
DFA Definition			All structs appropriately defined	
			20	
I/O			Program Input and Output appropriately defined	
			20	
C Constructs		All C constructs appropriately demonstrated (pointers, struct, malloc, sizeof, printf, functions, prototypes)	Most C constructs appropriately demonstrated	
		20 – 16	15	
Execution and	Program executes correctly for all DFA test cases	Program executes correctly for most DFA test cases	Program executes correctly for the given DFA	
	30 – 21	20 – 16	15	
Analysis	Correct Big-Oh Analysis	Reasonable Big-Oh analysis		
	10 – 6	5 – 1		
Deductions	Submitted on time Appropriately commented Executes using DevC++	Inappropriate comments 1-10% Compiles correctly	3% deducted per day late	**Not** submitted within six days of due date or **does not** compile