Finite Automata

The textbook describes implementation of a Finite Automata as a while loop with nested switch statements (pages 165 – 166). This is the most "direct" way to implements such a system with respect to the design (code) matching the process. But, such an approach means that every new Finite Automata requires a redesign/recode of the software. As discussed in class, there is another approach: table driven. This approach has advantages:

1. The resulting code runs faster
2. The code can be reconfigured with the change of a data structure, no redesign/recode is necessary

The disadvantage is that the table-driven approach requires more memory, the size of the table.

Assignment

Part I

1. Write a program to implement a Finite Automata using the table-driven approach.
2. Create the required table to represent the Finite Automata of Exercise 6.2.6, problem 1 on page 166 of the textbook and use the program to answer parts a – h of that problem.

Part II

1. Convert the following regular expression to recognize Java variable names to a Finite Automata diagram:

```
[a-zA-Z$_][a-zA-Z0-9$_]*
```

2. Using the program of Part I (above), create a table to represent a Finite Automata that recognizes Java variable names.
3. Use the program to determine if the following are legal Java variable names
    a. Temp123
    b. temp123
    c. $temp_variable
    d. 9temp_variable$

Deliverables

- Source code as specified above
- Essay (PDF format) including
    - Degree of success achieved

- The answers to the specified questions
- The Finite Automata diagram for Java variable names