

**Department of Computer Science
University of Massachusetts Lowell
COMP.3040
Fall 2019**

**Programming Assignment 1 [15 points]
Handed out on 10/6/2019
Due on 11/10/2019**

Implement a DFA simulator (in C, C++, Java or Python) in a Linux environment:

- Read in a specified machine (5-tuple definition) and process input strings against this DFA; output ACCEPT or NOT ACCEPT for each input string.
- All error checking must be performed including command line argument parsing, valid definition file, valid input string etc.
- Full help usage must be provided which should be self sufficient to run the program.
- Input string is read from stdin which means following should work
 - `./dfa -d m1.dfa <m1.in`
 - `./dfa -d m1.dfa <m1.in >m1.out`
 - `cat m1.in | ./dfa -d m1.dfa`
- -v option should provide verbose information on the working of the machine; for example display the definition of the machine, transitions as input is processed etc.

Deliverables

- Source files
- Sample Input/output
- Define 3 machines: 1.6 b, c, f [Homework #2] and show at least 3 ACCEPT and 3 NOT ACCEPT examples for each.
- Define machine for $a^*.c$ —as done in class; show at least 3 ACCEPT and 3 NOT ACCEPT examples
- You can show other examples too.
- 1 page report : Write about issues faced, lessons learned, any remaining bugs etc.

Extra Credit

- 3 categories
 - small: 2-4 points
 - medium: 4-8 points
 - large: 8-12 points
- [2-4 points] in addition, support json definition for machine; read machine defined in json; also output json for machine definition in verbose mode;
- [8-12 points] implement NFA with the same functionality.
- [4-8 points] implement in another language—python or C++ or C
- [4-8 points] show visualization and draw the machine
- [4-8 points] accepts expressions of the form $a^*.c$ and dynamically build NFA/DFA definition
- think of other command line options which could be useful.
- any other functionality – please document in report and code.

Deadline and Late Submissions

- The assignment is due on the date specified above at 11:59:59 PM

- Each day late will incur a penalty of 5% of the grade for the assignment; for example, if the assignment is 3 days late, the maximum grade will be 85 out of 100—15 will be subtracted from whatever grade is assigned.

Sample Run

[The sample output below is my implementation and is provided as a reference; please feel free to change/improve this when you implement.]

Usage:

```
./dfa-h -d <dfafile> -v
```

This is a simple DFA Implementation. The DFA definition is specified via .dfa file; input string is read from stdin. In non-verbose mode, print ACCEPT or NOT ACCEPT as the case may be.

```
-h          print usage
-d <dfafile>  DFA definition file
-v          verbose mode; display machine definition, transitions etc.
```

Example:

```
./dfa -d m1.dfa
```

Example dfa definition file m1.dfa

```
# DFA M1 from Page 36 of ITC Text;
states: q1 q2 q3
alphabet: 0 1
startstate: q1
finalstate: q2
transition: q1 0 q1
transition: q1 1 q2
transition: q2 0 q3
transition: q2 1 q2
transition: q3 0 q2
transition: q3 1 q2
```

Example run in interactive mode:

```
$ ./dfa -d m1.dfa
00011
00011 --> ACCEPT
00100
00100 --> ACCEPT
00000
00000 --> NOT ACCEPT
00000 --> NOT ACCEPT
```

Interactive Run:

```
$ ./dfa -d m1.dfa
00000
00000 --> NOT ACCEPT
11111
11111 --> ACCEPT
01010
01010 --> NOT ACCEPT
00100
00100 --> ACCEPT
00201
Invalid alphabet: 2; IGNORING rest of input
```

```

00201 --> NOT ACCEPT
11100
11100 --> ACCEPT
00a11
Invalid alphabet: a; IGNORING rest of input
00a11 --> NOT ACCEPT
110011
110011 --> ACCEPT
110011 --> ACCEPT

```

Use of Pipe and Redirection for input:

```

$ cat m1.in
00000
11111
00100
001001
001000
0010001

$ cat m1.in | ./dfa -d m1.dfa
00000 --> NOT ACCEPT
11111 --> ACCEPT
00100 --> ACCEPT
001001 --> ACCEPT
001000 --> NOT ACCEPT
0010001 --> ACCEPT
0010001 --> ACCEPT

$ ./dfa -d m1.dfa <m1.in
00000 --> NOT ACCEPT
11111 --> ACCEPT
00100 --> ACCEPT
001001 --> ACCEPT
001000 --> NOT ACCEPT
0010001 --> ACCEPT
0010001 --> ACCEPT

```

Interactive Run with -v flag:

```

$ ./dfa -d m1.dfa -v
---BEGIN DFA definition---
States:
  q1 q2 q3
Alphabet:
  0 1
StartState: q1
FinalState:
  q2
Transitions:
  q1 0 q1
  q1 1 q2
  q2 0 q3
  q2 1 q2
  q3 0 q2
  q3 1 q2

```

---END DFA definition---

001100

Current State: q1 Symbol: 0 -> New State: q1

Current State: q1 Symbol: 0 -> New State: q1

Current State: q1 Symbol: 1 -> New State: q2

Current State: q2 Symbol: 1 -> New State: q2

Current State: q2 Symbol: 0 -> New State: q3

Current State: q3 Symbol: 0 -> New State: q2

001100 --> ACCEPT

Current State: q1 Symbol: 0 -> New State: q1

Current State: q1 Symbol: 0 -> New State: q1

Current State: q1 Symbol: 1 -> New State: q2

Current State: q2 Symbol: 1 -> New State: q2

Current State: q2 Symbol: 0 -> New State: q3

Current State: q3 Symbol: 0 -> New State: q2

001100 --> ACCEPT

Use of Pipe and Redirection for input with -v flag:

```
$ ./dfa -d m1.dfa -v <m1.in >m1.out
```

```
$
```

```
$ cat m1.out
```

---BEGIN DFA definition---

States:

q1 q2 q3

Alphabet:

0 1

StartState: q1

FinalState:

q2

Transitions:

q1 0 q1

q1 1 q2

q2 0 q3

q2 1 q2

q3 0 q2

q3 1 q2

---END DFA definition---

Current State: q1 Symbol: 0 -> New State: q1

Current State: q1 Symbol: 0 -> New State: q1

Current State: q1 Symbol: 0 -> New State: q1

Current State: q1 Symbol: 0 -> New State: q1

Current State: q1 Symbol: 0 -> New State: q1

00000 --> NOT ACCEPT

Current State: q1 Symbol: 1 -> New State: q2

Current State: q2 Symbol: 1 -> New State: q2

Current State: q2 Symbol: 1 -> New State: q2

Current State: q2 Symbol: 1 -> New State: q2

Current State: q2 Symbol: 1 -> New State: q2

11111 --> ACCEPT

Current State: q1 Symbol: 0 -> New State: q1

Current State: q1 Symbol: 0 -> New State: q1

Current State: q1 Symbol: 1 -> New State: q2

Current State: q2 Symbol: 0 -> New State: q3

Current State: q3 Symbol: 0 -> New State: q2
00100 --> ACCEPT
Current State: q1 Symbol: 0 -> New State: q1
Current State: q1 Symbol: 0 -> New State: q1
Current State: q1 Symbol: 1 -> New State: q2
Current State: q2 Symbol: 0 -> New State: q3
Current State: q3 Symbol: 0 -> New State: q2
Current State: q2 Symbol: 1 -> New State: q2
001001 --> ACCEPT
Current State: q1 Symbol: 0 -> New State: q1
Current State: q1 Symbol: 0 -> New State: q1
Current State: q1 Symbol: 1 -> New State: q2
Current State: q2 Symbol: 0 -> New State: q3
Current State: q3 Symbol: 0 -> New State: q2
Current State: q2 Symbol: 0 -> New State: q3
001000 --> NOT ACCEPT
Current State: q1 Symbol: 0 -> New State: q1
Current State: q1 Symbol: 0 -> New State: q1
Current State: q1 Symbol: 1 -> New State: q2
Current State: q2 Symbol: 0 -> New State: q3
Current State: q3 Symbol: 0 -> New State: q2
Current State: q2 Symbol: 0 -> New State: q3
Current State: q3 Symbol: 1 -> New State: q2
0010001 --> ACCEPT
Current State: q1 Symbol: 0 -> New State: q1
Current State: q1 Symbol: 0 -> New State: q1
Current State: q1 Symbol: 1 -> New State: q2
Current State: q2 Symbol: 0 -> New State: q3
Current State: q3 Symbol: 0 -> New State: q2
Current State: q2 Symbol: 0 -> New State: q3
Current State: q3 Symbol: 1 -> New State: q2
0010001 --> ACCEPT