SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Engineering

Name: _	Date of Performance: _ _ /_ _/20_ _
Class: B.E. Computer	Date of Completion: _ _ /_ _/20_ _
Division :	Grade:
Batch:	
Roll No:	Sign. of Teacher with Date:
Subject: Advanced Computer Network Lab	

Experiment No. 6

Aim: Implementation of Deterministic Finite Automaton.

1. Objective: To study and demonstrate the concept of deterministic finite automata.

2. **Background:**

Finite automata is a state machine that takes a string of symbols as input and changes its state accordingly. Finite automata is a recognizer for regular expressions. When a regular expression string is fed into finite automata, it changes its state for each literal. If the input string is successfully processed and the automata reaches its final state, it is accepted, i.e., the string just fed was said to be a valid token of the language in hand.

The mathematical model of finite automata consists of:

- Finite set of states (Q)
- Finite set of input symbols (Σ)
- One Start state (q0)
- Set of final states (qf)
- Transition function (δ)

The transition function (δ) maps the finite set of state (Q) to a finite set of input symbols (Σ), $Q \times \Sigma \rightarrow Q$

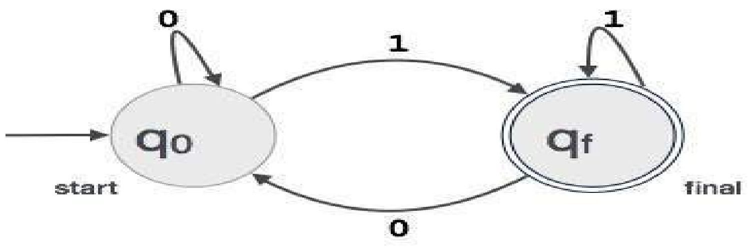
Finite Automata Construction

Let L(r) be a regular language recognized by some finite automata (FA).

- **States** : States of FA are represented by circles. State names are written inside circles.
- **Start state** : The state from where the automata starts, is known as the start state. Start state has an arrow pointed towards it.
- **Intermediate states** : All intermediate states have at least two arrows; one pointing to and another pointing out from them.
- **Final state** : If the input string is successfully parsed, the automata is expected to be in this state. Final state is represented by double circles. It may have any odd number of arrows pointing to it and even number of arrows pointing out from it. The number of odd arrows are one greater than even, i.e. **odd = even+1**.
- **Transition** : The transition from one state to another state happens when a desired symbol in the input is found. Upon transition, automata can either move to the next

state or stay in the same state. Movement from one state to another is shown as a directed arrow, where the arrows points to the destination state. If automata stays on the same state, an arrow pointing from a state to itself is drawn.

Example : We assume FA accepts any three digit binary value ending in digit 1. FA = {Q(q_0, q_f), Σ(0,1), q_0, q_f, δ}



3. Pre-lab Task:

Step1. State identification using lex
Step2. Prepare production rule for problem e.g. any number divisible by 3
Sep3. Implement parser program (YACC) for step 2.

4. In-lab Task:
Template for LEX program:

```
%{
--------------------
%}

%%
[--------------------] {return ZERO;}
[--------------------] {return ONE;}
[--------------------] {return TWO;}
[\n] {return NL;}
. ;
%%
```

Template for YACC Program :

```
%{
        --------------------
%}

%token ----     ------    -----     -----
%start q0

%%
```

```
q0 : ------ q0 {$$ = $2;}
        |        ------q1 {$$ = $2;}
        |        ------- q2 {$$ = $2;}
        | NL {printf("Number is divisible by 3\n"); return;}
        ;
q1 : ------  q1 {$$ = $2;}
        | ------  q2 {$$ = $2;}
        | ------  q0 {$$ = $2;}
        ;

q2: ------  q2 {$$ = $2;}
        | ------ q0 {$$ = $2;}
        | ------  q1 {$$ = $2;}
        ;

%%

main()
{
     yyparse();
}
```

Command for Execution:

```
lex file_name.l
yacc -d file_name.y
cc lex.yy.c y.tab.c -ll -ly
./a.out
```

5. Post-lab Task:

Outcomes:

. . . .
. . . .
. . . .
. . . .

Questions:

Q.1. Give the Difference between NFA & DFA.