Name: Naseem Shah	**Roll No:** 058
Exp. No: 01	**Date:**07/09/2021

DFA Implementation of (a/b)*abb

Aim: Write a C program to implement a DFA for the regular expression (a/b)*abb using IF-ELSE.

```
/*
C program to implement a DFA for the regular expression (a/b)*abb
using IF-ELSE.
*/

#include <stdio.h>
#include <stdbool.h>
#include <string.h>

int state0(char ch){
    if(ch == 'a')
        return 1;
    else if(ch == 'b')
        return 0;
    else
        return -1;
}

int state1(char ch){
    if(ch == 'a')
        return 1;
    else if(ch == 'b')
        return 2;
    else
        return -1;
}

int state2(char ch){
    if(ch == 'a')
        return 1;
    else if(ch == 'b')
        return 3;
    else
        return -1;
}

int state3(char ch){
    if(ch == 'a')
        return 1;
    else if(ch == 'b')
        return 0;
    else
        return -1;
}
```

```c
void main() {
    char s[1000];
    bool valid = true;
    int state = 0, final_state = 3;
    printf("input string : ");
    scanf("%s", s);

    for(int i = 0; s[i] != '\0' && valid; ++i){
        switch(state){
            case 0:
                state = state0(s[i]);
                break;
            case 1:
                state = state1(s[i]);
                break;
            case 2:
                state = state2(s[i]);
                break;
            case 3:
                state = state3(s[i]);
                break;
            default:
                valid = false;
                break;
        }
    }

    if(state != final_state)
        valid = false;

    if(valid)
        printf("Valid string!\n");
    else
        printf("Invalid string!\n");

    return;
}
```

Result: Successfully written C program to implement a DFA for the regular expression (a/b)*abb using IF-ELSE

Remarks:(To be filled by faculty)

Algorithm

1. Start
2. Create a NFA and convert it to a DFA, (or directly create a DFA) for the given regular expression (a/b)*abb
3. Read the input string, s
4. Set state = 0, final_state = 3, valid = true,
5. for each character, ch, in s, do
    a. if ch != ‘a’ and ch != ‘b’, set valid = false, and exit from loop

b. if state == 0, then do
    i. if ch == ‘a’, state = 1
    ii. if ch == ‘b’, state = 0
c. else if state == 1, then do
    i. if ch == ‘a’, state = 1
    ii. if ch == ‘b’, state = 2
d. else if state == 2, then do
    i. if ch == ‘a’, state = 1
    ii. if ch == ‘b’, state = 3
e. else if state == 3, then do
    i. if ch == ‘a’, state = 1
    ii. if ch == ‘b’, state = 0

6. if state != final_state, then, valid = false
7. if valid == true, then print “Valid string”, else print “Invalid string”
8. Stop

Diagrams & Tables

NFA

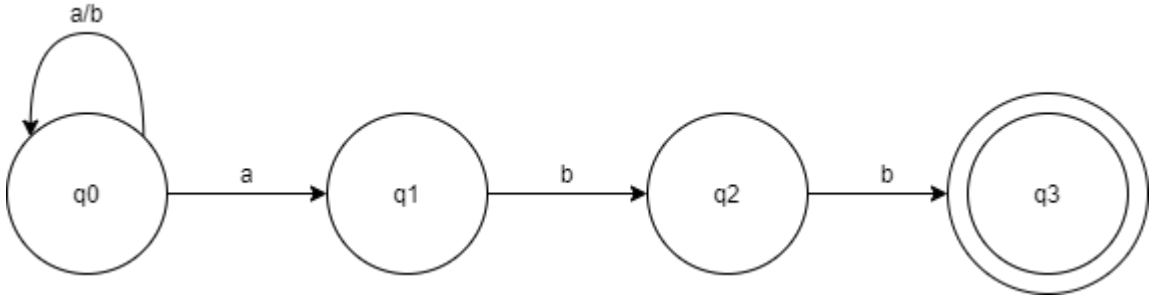


Table for NFA

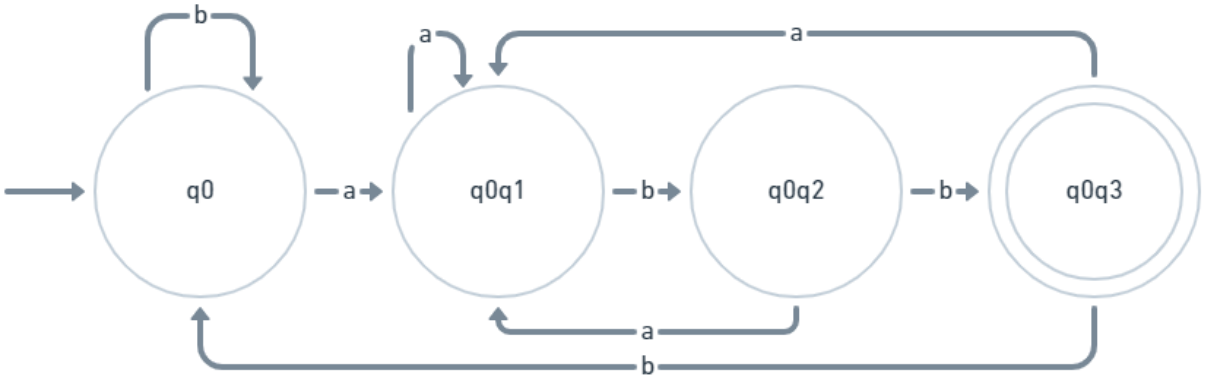
State	a	b
q0	{q0, q1}	q0
q1	ϕ	q2
q2	ϕ	q3
q3	ϕ	ϕ

Table for DFA

State	a	b
q0	[q0q1]	q0
[q0q1]	[q0q1]	[q0q2]
[q0q2]	[q0q1]	[q0q3]

[q0q3]	[q0q1]	q0

DFA



Sample output:

```
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP1# ls
exp1.c
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP1# gcc exp1.c
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP1# ./a.out
input string : hello
Invalid string!
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP1# ./a.out
input string : abab
Invalid string!
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP1# ./a.out
input string : abababbbbaabbbabb
Valid string!
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP1#
```