Name: Naseem Shah	**Roll No:** 058
Exp. No: 04	**Date:** 07/09/2021

DFA Implementation of (a+b+c)*abc(a+b+c)*

Aim: Write a C program to implement a DFA accepting strings made of {a,b,c} having ‘abc’ as a substring.

```c
/*
C program to implement a DFA accepting strings made of {a,b,c} having
'abc' as a substring.
*/

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

const int transition_table[][3] = {
    {1, 0, 0},      // state 0, initial state
    {1, 2, 0},      // state 1
    {1, 0, 3},      // state 2
    {3, 3, 3},      // state 3, final state
    {4, 4, 4}       // dead state
};
const int final_state[] = {3};
const int num_final_states =
sizeof(final_state)/sizeof(final_state[0]);

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

    for(int i=0; s[i] != '\0' ; i++){
        if(s[i] != 'a' && s[i] != 'b' && s[i] != 'c')
            state = 4;

        if(state == 4)
            break;
        state = transition_table[state][s[i] - 'a'];
    }

    for(int i=0; i<num_final_states; i++)
        if(state == final_state[i]){
            valid = true;
            break;
        }

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

    return;
}
```

Result: Successfully written C program to implement a DFA accepting strings made of {a,b,c} having ‘abc’ as a substring.

Remarks:(To be filled by faculty)

Algorithm

```
1.  Start
2.  Create a NFA, and then DFA for the given regular expression,
    (a+b+c)*abc(a+b+c)*
3.  Create transition table, transition_table[][], for the DFA obtained where each
    transition_table[i][j] denotes the current state i, and the next state when input is j.
    transition_table[4][3]= { {1,0,0}, {1,2,0}, {1,0,3}, {3,3,3}, {4,4,4} }
4.  Set final_states = {3}
5.  Read the input string, s
6.  Set state = 0, valid = false
7.  for each character ch in s, do
        a. if ch != ‘a’ and ch != ‘b’ and ch != ‘c’, then state = 4
        b. if state = 4, then break
        c. state = transition_table[state][ch - ‘a’]
8.  for i in final_states, do
        a. if i == state, then
            i.  valid = true
            ii. break
9.  if valid == true, then print “Valid string”, else print “Invalid string”
10. Stop
```

Diagrams & Tables

NFA

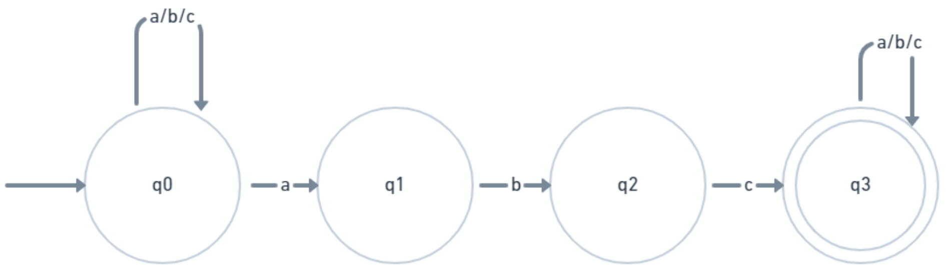


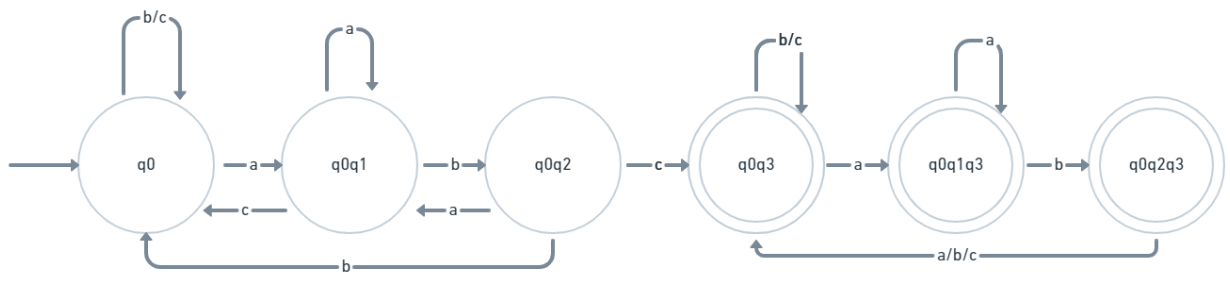
Table for NFA

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

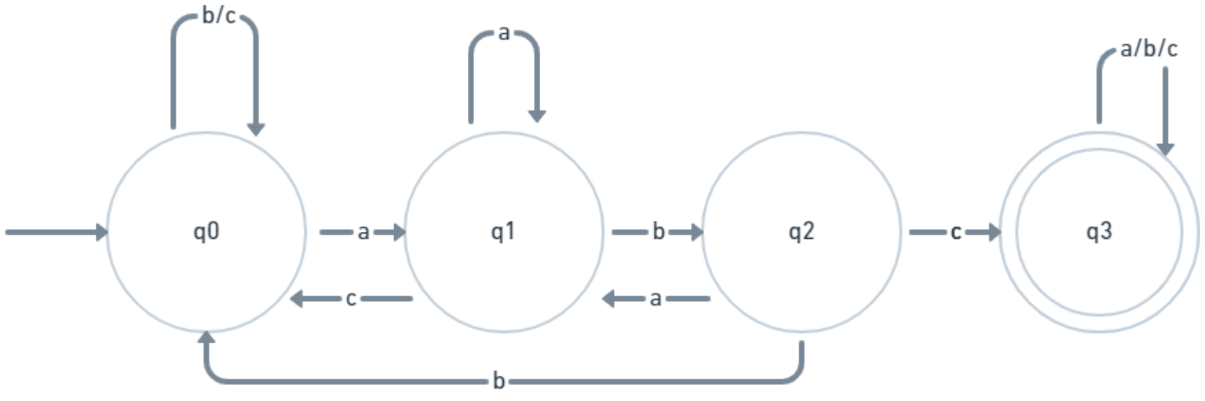
Table for DFA

State	a	b	c
q0	[q0q1]	q0	q0
[q0q1]	[q0q1]	[q0q2]	q0
[q0q2]	[q0q1]	q0	[q0q3]
*[q0q3]	[q0q1q3]	[q0q3]	[q0q3]
*[q0q1q3]	[q0q1q3]	[q0q2q3]	[q0q3]
*[q0q2q3]	[q0q3]	[q0q3]	[q0q3]

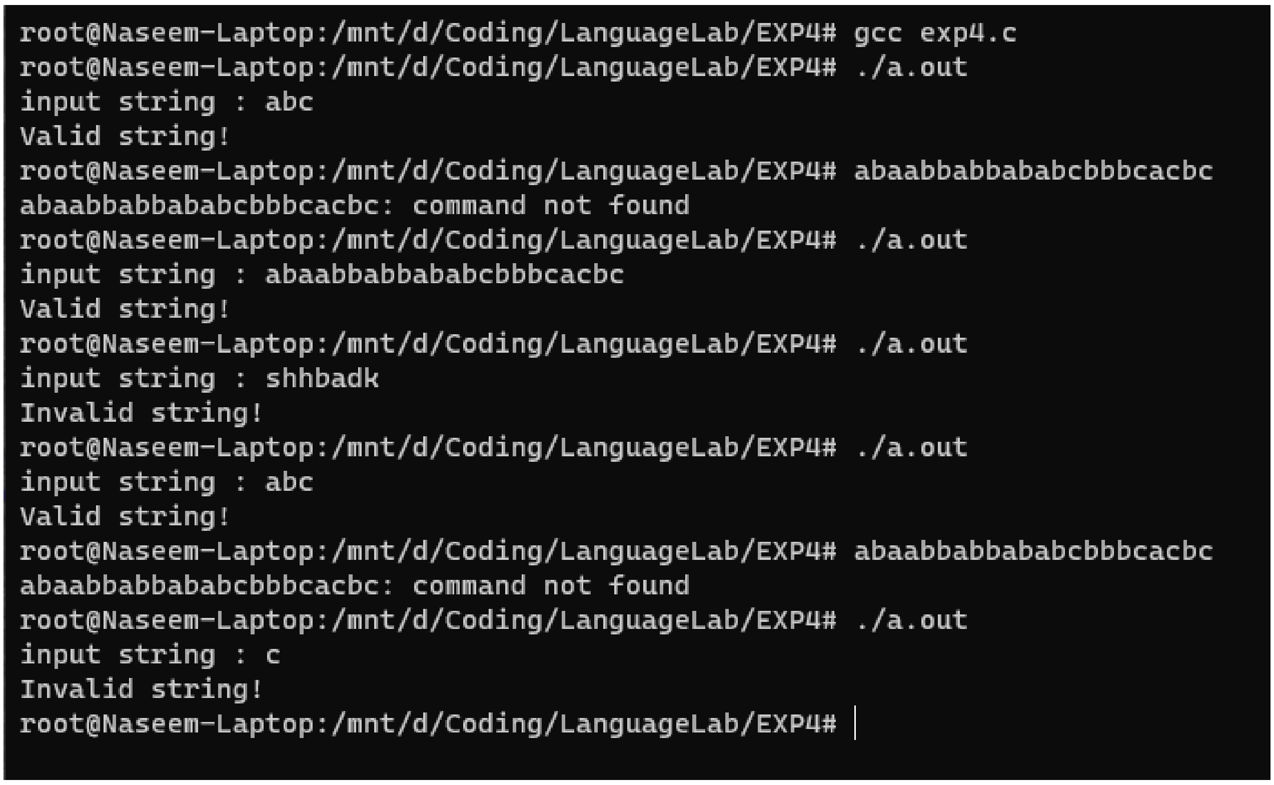
DFA



Minimized DFA



Sample output



```
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP4# gcc exp4.c
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP4# ./a.out
input string : abc
Valid string!
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP4# abaabbabbababcbbbcacbc
abaabbabbababcbbbcacbc: command not found
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP4# ./a.out
input string : abaabbabbababcbbbcacbc
Valid string!
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP4# ./a.out
input string : shhbadk
Invalid string!
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP4# ./a.out
input string : abc
Valid string!
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP4# abaabbabbababcbbbcacbc
abaabbabbababcbbbcacbc: command not found
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP4# ./a.out
input string : c
Invalid string!
root@Naseem-Laptop:/mnt/d/Coding/LanguageLab/EXP4#
```