Name – Bhargav Shamuvel Gurav **PRN** – 2041009

Class – L.Y. B-Tech (Computer) **Batch** – B1

Course Code – CO406U **Course Name -** CDL

Practical no. 3

Aim: Write a program to recognize strings under ‘a*’, ‘a*b+’, ‘abb’.

Theory :

Automata – What is it?

The term "Automata" is derived from the Greek word "αὐτόματα" which means "self-acting". An automaton (Automata in plural) is an abstract self-propelled computing device which follows a predetermined sequence of operations automatically.

An automaton with a finite number of states is called a Finite Automaton (FA) or Finite State Machine (FSM).

Formal definition of a Finite Automaton

An automaton can be represented by a 5-tuple (Q, ∑, δ, q0, F), where −

- Q is a finite set of states.
- ∑ is a finite set of symbols, called the alphabet of the automaton.
- δ is the transition function.
- q0 is the initial state from where any input is processed (q0 ∈ Q).
- F is a set of final state/states of Q (F ⊆ Q).

Related Terminologies

Alphabet

- Definition − An alphabet is any finite set of symbols.
- Example − ∑ = {a, b, c, d} is an alphabet set where ‘a’, ‘b’, ‘c’, and ‘d’ are symbols.

String

- Definition − A string is a finite sequence of symbols taken from ∑.
- Example − ‘cabcad’ is a valid string on the alphabet set ∑ = {a, b, c, d}

Length of a String

- Definition − It is the number of symbols present in a string. (Denoted by |S|).
- Examples −
  - If S = ‘cabcad’, |S|= 6

- If $|S| = 0$, it is called an empty string (Denoted by λ or ε)

Kleene Star

- Definition − The Kleene star, $\sum^*$, is a unary operator on a set of symbols or strings, $\sum$, that gives the infinite set of all possible strings of all possible lengths over $\sum$ including λ.
- Representation − $\sum^* = \sum 0 \cup \sum 1 \cup \sum 2 \cup \ldots\ldots.$ where $\sum p$ is the set of all possible strings of length p.
- Example − If $\sum = \{a, b\}$, $\sum^* = \{\lambda, a, b, aa, ab, ba, bb, \ldots\ldots\ldots\ldots.\}$

Kleene Closure / Plus

- Definition − The set $\sum+$ is the infinite set of all possible strings of all possible lengths over $\sum$ excluding λ.
- Representation − $\sum+ = \sum 1 \cup \sum 2 \cup \sum 3 \cup \ldots\ldots.$
  $\sum+ = \sum^* - \{ \lambda \}$
- Example − If $\sum = \{ a, b \}$, $\sum+ = \{ a, b, aa, ab, ba, bb, \ldots\ldots\ldots\ldots \}$

Language

- Definition − A language is a subset of $\sum^*$ for some alphabet $\sum$. It can be finite or infinite.
- Example − If the language takes all possible strings of length 2 over $\sum = \{a, b\}$, then $L = \{ ab, aa, ba, bb \}$

Deterministic Finite Automaton

Finite Automaton can be classified into two types −

- Deterministic Finite Automaton (DFA)
- Non-deterministic Finite Automaton (NDFA / NFA)

Deterministic Finite Automaton (DFA)

In DFA, for each input symbol, one can determine the state to which the machine will move. Hence, it is called Deterministic Automaton. As it has a finite number of states, the machine is called Deterministic Finite Machine or Deterministic Finite Automaton.

Graphical Representation of a DFA

A DFA is represented by digraphs called state diagram.

- The vertices represent the states.
- The arcs labeled with an input alphabet show the transitions.
- The initial state is denoted by an empty single incoming arc.
- The final state is indicated by double circles.

Example

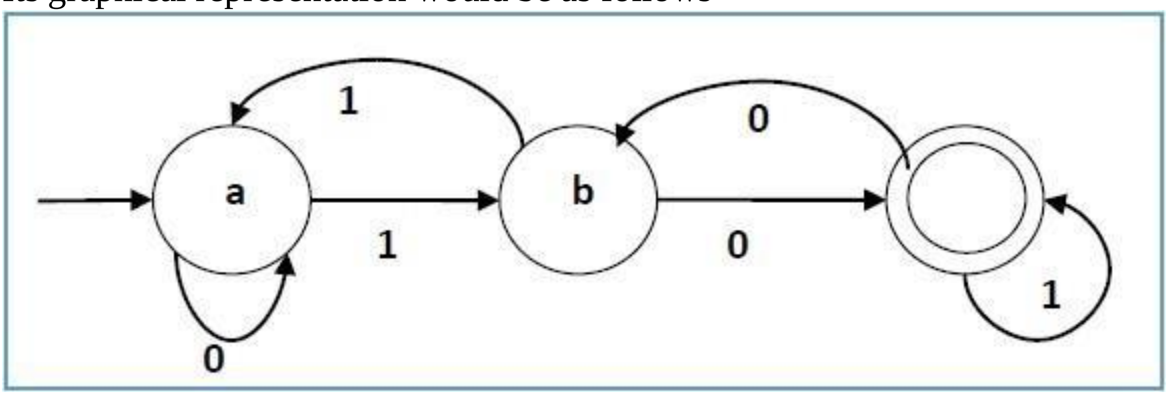
Let a deterministic finite automaton be →

- Q = {a, b, c},
- ∑ = {0, 1},
- q0 = {a},
- F = {c}, and

Transition function δ as shown by the following table −

Present State	Next State for Input 0	Next State for Input 1
a	a	b
b	c	a
c	b	c

Its graphical representation would be as follows −



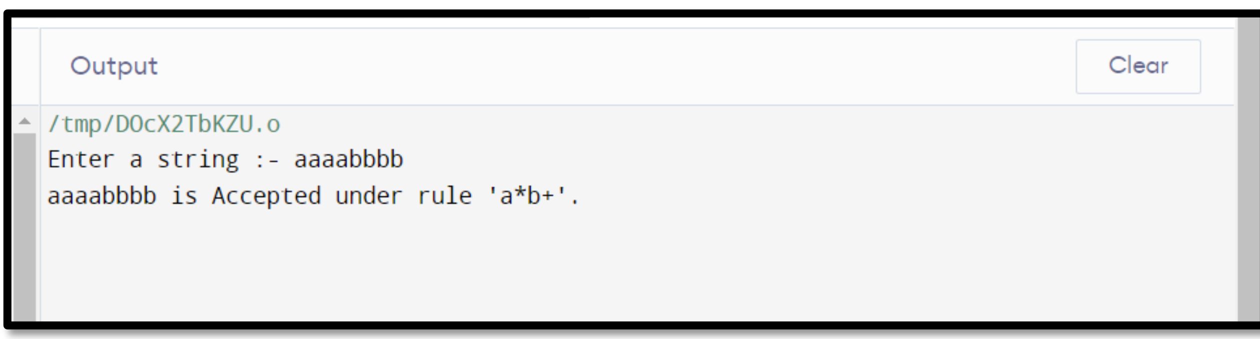
Program Code:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
	char s[20],c;
	int state=0,i=0;
	printf("\n Enter a string :- ");
	gets(s);
	while(s[i]!='\0')
	{
		switch(state)
		{
```

```
case 0: c=s[i++];
        if(c=='a')
                state=1;
        else if(c=='b')
                state=2;
        else
                state=6;
break;
case 1: c=s[i++];
        if(c=='a')
                state=3;
        else if(c=='b')
                state=4;
        else
                state=6;
break;
case 2: c=s[i++];
        if(c=='a')
                state=6;
        else if(c=='b')
                state=2;
        else
                state=6;
break;
case 3: c=s[i++];
        if(c=='a')
                state=3;
        else if(c=='b')
                state=2;
        else
                state=6;
break;
case 4: c=s[i++];
        if(c=='a')
                state=6;
        else if(c=='b')
                state=5;
        else
                state=6;
break;
```

```
                        case 5: c=s[i++];
                                if(c=='a')
                                        state=6;
                                else if(c=='b')
                                        state=2;
                                else
                                        state=6;
                                break;
                        case 6: printf("\n %s is not Recognised By the Automata.",s);
                exit(0);
                }
        }
if((state==1)||(state==3))
        printf("\n %s is Accepted under rule 'a*'.",s);
else if((state==2)||(state==4))
        printf("\n %s is Accepted under rule 'a*b+'.",s);
else if(state==5)
        printf("\n %s is Accepted under rule 'abb'.",s);
}
```

Output:



```
Output                                                    Clear
/tmp/DOcX2TbKZU.o

 Enter a string :- abb
 abb is Accepted under rule 'abb'.
```

```
Output                                                    Clear
/tmp/DOcX2TbKZU.o
Enter a string :- abbbb
abbbb is Accepted under rule 'a*b+'.
```

```
Output                                                    Clear
/tmp/DOcX2TbKZU.o
Enter a string :- aabbaabba
aabbaabba is not Recognised By the Automata.
```

Conclusion : In this practical we learnt how DFA recognizes patterns for identifiers, keywords.