SSGMCE	SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGG.	LABORATORY MANUAL
	PRACTICAL EXPERIMENT INSTRUCTION SHEET	
	EXPERIMENT TITLE : Write a C program to recognize strings under 'a', 'a*b+', 'abb'.	
EXPERIMENT NO. : SSGMCE/WI/IT/01/6IT01/03	ISSUE NO. : 00	ISSUE DATE : 01.02.2022
REV. DATE :	REV. NO. :	DEPTT. : INFORMATION TECHNOLOGY
LABORATORY : COMPILER DESIGN (CD)	SEMESTER : VI	PAGE: OF 4

1.0) AIM:

Write a C program to recognize strings under 'a', 'a*b+', 'abb'.

2.0) OBJECTIVE:

After the completion of this experiment, lexical analyzer will be able to recognize strings under 'a*', 'a*b+', 'abb'.

3.0) FACILITIES/ APPARATUS:

i) Hardware : Computer Machine

ii) Software : Turbo C++

4.0) THEORY:

The lexical analyzer needs to scan and identify only a finite set of valid string/token/lexeme that belong to the language in hand. It searches for the pattern defined by the language rules. Regular expressions have the capability to express finite languages by defining a pattern for finite strings of symbols. The grammar defined by regular expressions is known as **regular grammar**. The language defined by regular grammar is known as **regular language**.

Regular expression is an important notation for specifying patterns. Each pattern matches a set of strings, so regular expressions serve as names for a set of strings. A regular expression is a sequence of characters that define a pattern hence tokens can be described by regular languages. Regular languages are easy to understand and have efficient implementation. Various algebraic laws that obeyed by regular expressions, can be used to manipulate regular expressions into its equivalent forms.

1. One or more instances called as Positive Closure : +
2. Zero or more instances called as Kleene Closure : *

5.0) PROGRAM:

```
#include<stdio.h>
#include<conio.h>
 #include<string.h>
#include<stdlib.h>
void main()
```

PREPARED BY:
PROF. S. D. PADIYA

APPROVED BY:(H.O.D.)
PROF. A. S. MANEKAR

```
{
        char s[20], c;
        int state=0, i=0;
        clrscr();
        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.", s);
exit(0);
    }
}
```

```
if(state==1)
        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);
    getch();
}
```

6.0) OUTPUT OF PROGRAM

Enter a string:

INPUT

a

OUTPUT

a is accepted under rule 'a'

INPUT

aaaaabb

OUTPUT

aaaaabb is accepted under rule 'a*b+'

INPUT

abcd

OUTPUT

abcd is not recognized.

7.0) CONCLUSION:

A lexical analyzer has been designed using C language for the given language in which it recognized the strings under 'a', 'a*b+' and 'abb'.