EXPERIMENT-1

PROBLEM: Write a C program to identify whether a given line is a comment or not.

PROGRAM LOGIC: Read the input string.

Check whether the string is starting with '/' and check next character is '/' or'*'.

If condition satisfies print comment.

Else not a comment.

Solution:

```
#include<stdio.h>
Int main()
{
        char com[30];
        int i=2,a=0;
        printf("\n Enter comment:");
        gets(com);
        if(com[0]=='/')
        {
                if(com[1]=='/')
                        printf("\n It is a comment");
                else if(com[1]=='*')
                {
                        for(i=2;i<=30;i++)
                        {
                                if(com[i]=='*'&&com[i+1]=='/')
                                {
                                        printf("\n It is a comment");
                                        a=1;
                                        break;
                                }
                                else
                                        continue;
                        }
                        if(a==0)
                                printf("\n It is not a comment");
                }
                else
                        printf("\n It is not a comment");
        }
        else
        printf("\n It is not a comment");
return 0;}
```

INPUT & OUTPUT:

Input: Enter comment: //hello

Output: It is a comment

Input: Enter comment: hello

Output: It is not a comment

EXPERIMENT-2

PROBLEM: Write a C program to test whether a given identifier is valid or not.

PROGRAM LOGIC: Read the given input string.

Check the initial character of the string is numerical or any special character except '_' then print it is not a valid identifier.

Otherwise print it as valid identifier if remaining characters of string doesn't contains any special characters except '_'.

PROGRAM:

```
#include<stdio.h>
#include<ctype.h>
int main()
{
        char a[10];
        int flag, i=1;
        printf("\n Enter an identifier:");
        gets(a);
        if(isalpha(a[0]))
                flag=1;
        else
                printf("\n Not a valid identifier");
        while(a[i]!='\0')
        {
                 if(!isdigit(a[i])&&!isalpha(a[i]))
                {
                        flag=0;
                         break;
                }
         i++;
        }
        if(flag==1)
                 printf("\n Valid identifier");
return 0;
}
```

INPUT & OUTPUT:

Input: Enter an identifier: first

Output:

```
        Valid identifier
        Enter an identifier:1aqw
        Not a valid identifier
```

EXPERIMENT-3

PROBLEM: Write a C program to simulate lexical analyzer for validating operators.

PROGRAM LOGIC: Read the given input.

If the given input matches with any operator symbol.

Then display in terms of words of the particular symbol.

Else print not a operator.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
int main()
{
char s[5];
printf("\n Enter any operator:");
gets(s);
switch(s[0])
{
case'>': if(s[1]=='=')
printf("\n Greater than or equal");
else
printf("\n Greater than");
break;
case'<': if(s[1]=='=')
printf("\n Less than or equal");
else
printf("\nLess than");
break;
case'=': if(s[1]=='=')
printf("\nEqual to");
else
printf("\nAssignment");
break;
case'!': if(s[1]=='=')
printf("\nNot Equal");
else
printf("\n Bit Not");
break;
case'&': if(s[1]=='&')
printf("\nLogical AND");
else
printf("\n Bitwise AND");
break;
case'|': if(s[1]=='|')
printf("\nLogical OR");
10
else
printf("\nBitwise OR");
break;
case'+': printf("\n Addition");
```

```
break;
case'-': printf("\nSubstraction");
break;
case'*': printf("\nMultiplication");
break;
case'/': printf("\nDivision");
break;
case'%': printf("Modulus");
break;
default: printf("\n Not a operator");
}
return 0;
}
```

INPUT & OUTPUT:

Input: Enter any operator: *

Output: Multiplication

EXPERIMENT-4

PROBLEM: To find whether given string is keyword or not

PROGRAM:

```
#include<stdio.h>
#include<string.h>
int main()
{
char a[5][10]={"printf","scanf","if","else","break"};
char str[10];
int i,flag;
clrscr();
puts("Enter the string :: ");
gets(str);
for(i=0;i<strlen(str);i++)
{
if(strcmp(str,a[i])==0)
{
flag=1;
break;
}
else
flag=0;
}
if(flag==1)
puts("Keyword");
else
puts("String");
return 0;
}
```

Output

```
Enter the string :: printf
Keyword
Enter the string :: vikas
String
```

EXPERIMENT-5

PROBLEM: To find whether given string is constant or not

PROGRAM:

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
int main()
{
int i,flag;
char a[5];
clrscr();
puts("Enter the value :: ");
gets(a);
for(i=0;i<strlen(a);i++)
{
if(isdigit(a[i]))
flag=1;
else
{
flag=0;
break;
}
}
if(flag==1)
puts("Value is constant");
else
puts("Value is a variable");
return 0;
}
```

Output

```
Enter the value ::
123
Value is constant
Enter the value ::
vikas
Value is a variable
```

EXPERIMENT-6

PROBLEM: To count blank space and count the number of lines

PROGRAM:

```
#include<stdio.h>
#include<string.h>
int main()
{
int flag=1;
char i,j=0,temp[100];
clrscr();
printf("Enter the Sentence (add '$' at the end) :: \n\n");
while((i=getchar())!='$')
{
if(i==' ')
i=';';
else if(i=='\t')
i='"';
else if(i=='\n')
flag++;
temp[j++]=i;
}
temp[j]=NULL;
printf("\n\n\nAltered Sentence :: \n\n");
puts(temp);
printf("\n\nNo. of lines = %d",flag);
return 0;
}
```

Output

```
Enter the Sentence (add '$' at the end) ::
vikas  kapoor
hello   world
welcome$

Altered Sentence::

vikas;kapoor
hello"world
welcome

No. of lines = 3
```

EXPERIMENT-7

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

PROGRAM LOGIC:

By using transition diagram we verify input of the state.
If the state recognize the given pattern rule.
Then print string is accepted under a*/ a*b+/ abb.
Else print string not accepted.

PROGRAM:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
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);
 return 0;
}
```

INPUT & OUTPUT:

Input :

```
 Enter a String: aaaabbbbb
```

Output:

```
 aaaabbbbb is accepted under rule 'a*b+'
```

Input :

```
 Enter a string: cdgs
```

Output:

```
 cdgs is not recognized
```

ASSIGNMENTS

EXPERIMENT-8

Assignments-1: Write a program for SHIFT REDUCE PARSER

EXPERIMENT-9

Assignments-2: Write a program to find the FIRST of a given grammar

EXPERIMENT-10

Assignments-3: Write a program to find the FOLLOW of a given grammar