Program-1

Aim:- Write a c program to identify whether a given line is a comment or not.

Program:-

```
#include <stdio.h>
#include <conio.h>
void main()
{
    char com[30];
    int i=2, a=0;
    clrscr();
    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");
    getch();
  }
```

output :-

\n Enter comment // hi neeru

It is a comment

\n Enter comment hi neeru

It is not a comment.

Result :- The above program is executed successfully.

Program - 2

Aim:- write a C program to test whether a given identifier is valid or not.

Program:-

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
void main()
{
    char a[10];
    int flag, i=1;
    clrscr();
    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("In valid Identifier");
  getch();
  }
```

output:-

Enter an identifier Neeru

valid Identifier.

Enter an identifier 1Neeru

Invalid identifier.

Result:- The above program is executed successfully.

Program-3

Aim:- Write a c program to simulate lexical analyser for validating operations.

Program:-

```
#include <stdio.h>
#include <conio.h>
Void main()
{
   char s[5];
   clrscr();
   printf("enter any operator");
   gets(s);
   Switch(s[0])
   {
    case '>': if(s[1]=='=')
    printf("in greater than or equal");
    Else
    printf("in greater than");
    break;
   Case '<': if(s[1]=='=')
   printf("in lessthan or equal");
    Else
   printf("in lessthan");
    break;
```

```
Case '=': if (S[i]=='=')
printf ("In Equal to");
Else
 printf ("In Assignment");
  break;
Case '!': if (S[i]=='=')
printf ("In not Equal to");
Else
printf("In Bit not");
 break;
Case '&': if (S[i]=='&')
 printf ("In logical AND");
 Else
 printf ("Bitwise AND");
  break;
 Case '|': if (S[i]== '|')
 printf ("In logical or");
 else
  printf("In Bitwise or");
  break;
Case '-': printf("In substraction");
   break;
Case '*': printf ("In multiplication");
```

```
    break;
  case '+': printf ("\n Addition");
    break;
  case '/': printf("\n division");
    break;
  case '%': printf("\n modulus");
    break;
  default: printf("\n not a operator");
  }
  getch();
}
```

output:-

Enter any operator &
bitwise and
Enter any operator =
Assignment.

Result:- The above program is executed successfully.

Program - 4

Aim:- Write a c program to recognize strings under 'a*', 'a*b+', 'abb'

Program:-

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
void 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);
getch();
}
```

output:-

```
Enter a string : aaabbb
aaabbb is accepted under rule 'a*b+'
Enter a string : abb
abb is accepted under rule 'abb'
```

Result:- The above program is executed successfully.

program-5

Aim:- Write a C program for constructing of LL(1) parser.

program:-

```
#include <stdio.h>
#include <conio.h>
#include<string.h>
char S[20], stack[20];
void main()
{
  char m[5][6][3]={"tb","","","tb","","","","","+tb",
"","","n","n","fc","","","fc","","","","n","*fc",
"a","n","n",";","","","(e)","",""};
  int size[5][6]={2,0,0,2,0,0,0,3,0,0,1,1,2,0,0,2,0,0,0,1,3,0,1,1,1,
                  0,0,3,0,0};
  int i,j,k,n,str1,str2;
  clrscr();
  printf("\n enter the input string");
  scanf("%s",s);
  strcat(s,"$");
  n=strlen(s);
  stack[0]='$';
  stack[1]='e';
```

```
    i=1;
    j=0;
printf("\n stack input \n");
printf ("------\n");
while ((stack[i]!='$') && (s[j]!='$'))
 {
   if (stack[i]==s[j])
   {
     i--;
     j++;
   }
   switch (stack[i])
   {
    case 'e': str1=0;
              break;
    case 'b': str1=1;
              break;
    case 't': str1=2;
              break;
    case 'c': str1=3;
              break;
    case 'f': str1=4;
              break;
   }
   switch (s[j])
   {
```

```
Case 'i' : str2=0;
           break;
Case '+' : str2=1;
           break;
Case '*' : str2=2;
           break;
Case '(' : str2=3;
           break;
Case ')' : str2=4;
           break;
Case '$' : str2=5;
           break;
}
if(m [str1][str2][0]=='\0')
{
 printf ("\n Error");
 exit(0);
}
Else if (m[str1][str2][0]=='n')
 i--;
Else if(m[str1][str2][0]=='i')
 stack[i]='i';
Else
{
 for (k=size[str1][str2]-1; k>=0; k--)
 {
```

```
        stack[i] = m[str1][str2][k];
        i++;
    }
    j--;
}
for(k=0; k<=i; k++)
printf("%c", stack[k])
printf(" ");
for(k=j; k<=n; k++)
 printf("%c", s[k]);
printf("\n");
}
printf("\n success");
 getch();
}
```

Output :-

Enter the input string: i*i+i

Stack	input
$bt	i*i+i$
$bcf	i*i+i$
$bci	i*i+i$
$bcf*	*i+i$
$bci	i+i$
$b	+i$
$bt+	+i$
$bcf	i$
$bci	i$
$b	$

Success.

Result:- The above program is executed successfully.

program - 6

Aim:- Construction of recursive descent parsing for the following grammer.

$E \rightarrow TE'$

$E' \rightarrow +TE/@$

$T \rightarrow FT'$

$T' \rightarrow *FT'/@$

$F \rightarrow (E)/ID$

"@ represents null character".

Program :-

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
 char input[100];
 int i,l;
 void main()
 {
  clrscr();
  printf("\n recursive descent parsing for the following
                                        grammer\n");
  printf("\n productions\n");
  printf("\n Enter the string to be checked");
  gets(input);
   if (E())
   {
```

```
    if(input[i+1]=='0')
    printf("\n String is accepted");
     else
    printf("\n string is not accepted");
  }
 else
 printf("\n string not accepted");
 getch();
 }
E()
{
  if(T())
  {
   if(Ep())
     return 1;
      else
      return(0);
  }
  else
     return(0);
   }
  Ep()
  {
    if(input[i]=='+')
     { i++;
```

```
if (T())
{
if (Ep())
return 1;
Else
return(0);
}
Else
return(0);
}
Else
return(0);
}
T()
{
if (F())
{
if (TP())
return(1);
Else
return(0);
}
Else
return(0);
}
TP()
{
if(input[i] == '*')
```

```
{
  i++;
  if (F())
  {
   if (TP())
    return(1);
    Else
     return(0);
    }
    Else
      return(0);
  }
  Else
    return(0);
 }
 F()
{
if (input[i]=='(')
  {
   i++;
 If (E())
 {
  if (input[i]==')')
    {
      i++;
      return(1);
    }
    Else
      return(0);
    }
```

```
}
Else
 return(a);
}
Else if( input[i] >= 'a' && input [i] <= 'z' || input[i] >= 'A' &&
     input [i] <= 'z')
{
  i++;
  return(1);
}
 Else
   return(0);
}
```

output :-

In recursive descent parsing for fig grammer in productions

$E \rightarrow TE'$

$E' \rightarrow TE'/@$

$T \rightarrow FT'$

$T' \rightarrow *FT'/@$

$F \rightarrow (E)/id$

enter the string to be checked

(a*b)+c

string not accepted

Result :- The above program is executed successfully.