EXPERIMENT -04

Aim:- Program to implement Predictive Parsing LL(1) in C

Code:-

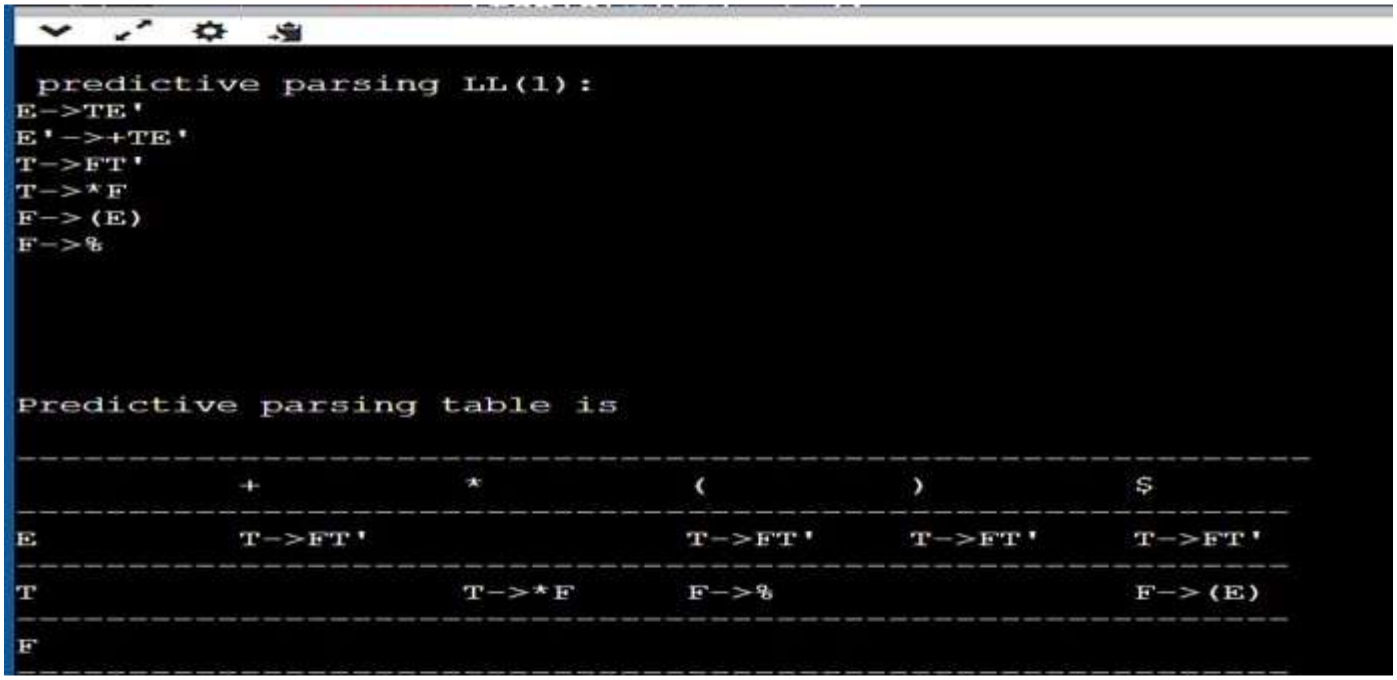
INPUT:-

```
#include<stdio.h>
#include<conio.h>
#include<string.h>

char prol[10][10]={"E","E'","E'","T","T'","T'","F","F"};
char pror[10][10]={"TE'","+TE'","@","FT'","*FT'","@","(E)","%"};
char prod[10][10]={"E->TE'","E'->+TE'","T->FT'","T->*F","F->(E)","F->%"};
char first[10][10]={"(%","+@","(%","*@","(%"};
char follow[10][10]={"$)","$)","+$)","+$)","*+$)"}; char table[5][6][10];
numr(char c)
{
   switch(c)
   {
      case 'E': return 0;
      case 'T': return 1;
      case 'F': return 2;
      case '+': return 0;
      case '*': return 1;
      case '(': return 2;
      case ')': return 3;
      case '%': return 4;
      case '$': return 5;
   }
   return(2);
}
   void main()
   {
      int i,j,k;
      // clrscr(); for(i=0;i<5;i++) for(j=0;j<6;j++)
```

```
strcpy(table[i][j]," ");
 printf("\n predictive parsing LL(1):\n");
 for(i=0;i<10;i++) printf("%s\n",prod[i]);
 printf("\nPredictive parsing table is\n");
 fflush(stdin);
 for(i=0;i<10;i++)
 {
     k=strlen(first[i]);
     for(j=0;j<10;j++)
     if(first[i][j]!='@')
     strcpy(table[numr(prol[i][0])+1][numr(first[i][j])+1],prod[i]);
 }
 for(i=0;i<10;i++)
 {
     if(strlen(pror[i])==1)
     {
         if(pror[i][0]=='@')
         {
             k=strlen(follow[i]);
             for(j=0;j<k;j++)
                 strcpy(table[numr(prol[i][0])+1][numr(follow[i][j])+1],prod[i]);
         }
     }
 }
 strcpy(table[0][0]," ");
 strcpy(table[0][1],"+");
 strcpy(table[0][2],"*");
 strcpy(table[0][3],"(");
 strcpy(table[0][4],")");
 strcpy(table[0][5],"%");
 strcpy(table[0][5],"$");
 strcpy(table[1][0],"E");
 strcpy(table[2][0],"T");
 strcpy(table[3][0],"F");
```

```
    printf("\n______________________________________________\n");
    for(i=0;i<5;i++)
    for(j=0;j<6;j++)
    {
        printf("%-10s",table[i][j]);
        if(j==5)
      printf("\n............................................................\n");
    }
   getch();
}
```

OUTPUT:-



```
 predictive parsing LL(1):
E->TE'
E'->+TE'
T->FT'
T->*F
F->(E)
F->%



Predictive parsing table is
-----------------------------------------------------------
          +         *         (         )         $
-----------------------------------------------------------
E         T->FT'              T->FT'    T->FT'    T->FT'
-----------------------------------------------------------
T                   T->*F     F->%                F->(E)
-----------------------------------------------------------
F
-----------------------------------------------------------
```

EXPERIMENT -05

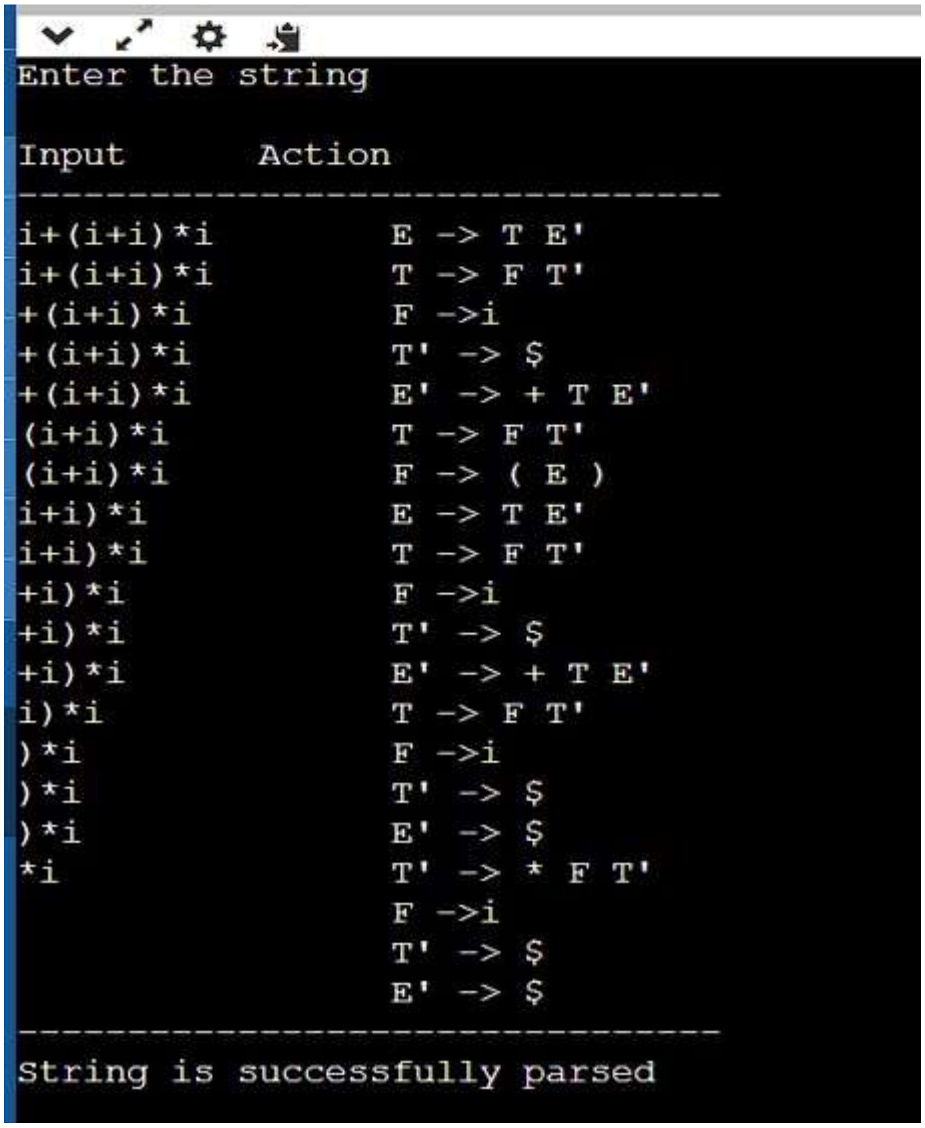
Aim:- Program to implement Recursive Descent Parsing in C.

Code:-

INPUT:-

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

#define SUCCESS 1
#define FAILED 0

int E(), Edash(), T(), Tdash(), F();

const char *cursor;
char string[64];

int main()
{
    puts("Enter the string");
    // scanf("%s", string);
    sscanf("i+(i+i)*i", "%s", string);
    cursor = string;
    puts("");
    puts("Input      Action");
    puts("................................................");

    if (E() && *cursor == '\0') {
        puts("................................................");
        puts("String is successfully parsed");
        return 0;
    } else {
        puts("................................................");
        puts("Error in parsing String");
        return 1;

    }
}

int E()
{
    printf("%-16s E -> T E'\n", cursor);
    if (T()) {
        if (Edash())
            return SUCCESS;
        else
            return FAILED;
    } else
        return FAILED;
}

int Edash()
{
    if (*cursor == '+') {
        printf("%-16s E' -> + T E'\n", cursor);
        cursor++;
        if (T()) {
            if (Edash())
                return SUCCESS;
            else
                return FAILED;
        } else
            return FAILED;
    } else {
        printf("%-16s E' -> $\n", cursor);
        return SUCCESS;
    }
}

int T()
{
    printf("%-16s T -> F T'\n", cursor);
    if (F()) {
        if (Tdash())
            return SUCCESS;
        else

       return FAILED;
    } else
        return FAILED;
}

int Tdash()
{
    if (*cursor == '*') {
        printf("%-16s T' -> * F T'\n", cursor);
        cursor++;
        if (F()) {
            if (Tdash())
                return SUCCESS;
            else
                return FAILED;
        } else
            return FAILED;
    } else {
        printf("%-16s T' -> $\n", cursor);
        return SUCCESS;
    }
}

int F()
{
    if (*cursor == '(') {
        printf("%-16s F -> ( E )\n", cursor);
        cursor++;
        if (E()) {
            if (*cursor == ')') {
                cursor++;
                return SUCCESS;
            } else
                return FAILED;
        } else
            return FAILED;
    } else if (*cursor == 'i') {
        cursor++;
        printf("%-16s F ->i\n", cursor);
        return SUCCESS;

  } else
        return FAILED;
}
```

OUTPUT:-



```
Enter the string

Input      Action
--------------------------------
i+(i+i)*i        E -> T E'
i+(i+i)*i        T -> F T'
+(i+i)*i         F ->i
+(i+i)*i         T' -> $
+(i+i)*i         E' -> + T E'
(i+i)*i          T -> F T'
(i+i)*i          F -> ( E )
i+i)*i           E -> T E'
i+i)*i           T -> F T'
+i)*i            F ->i
+i)*i            T' -> $
+i)*i            E' -> + T E'
i)*i             T -> F T'
)*i              F ->i
)*i              T' -> $
)*i              E' -> $
*i               T' -> * F T'
                 F ->i
                 T' -> $
                 E' -> $
--------------------------------
String is successfully parsed
```

EXPERIMENT -06

Aim:- Program to implement Operator Precedence Parsing in C.

Code:-

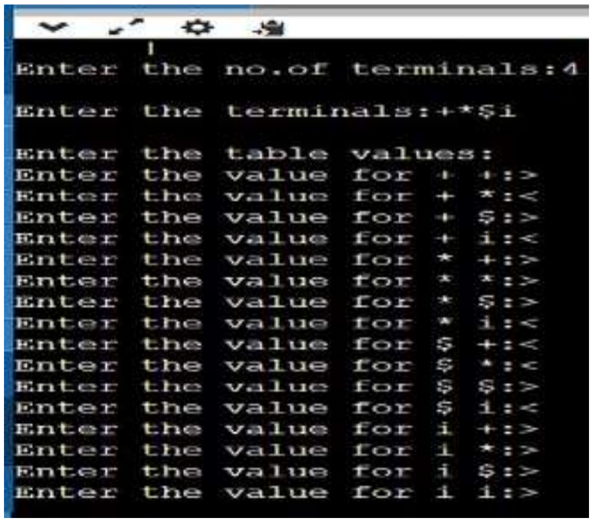
INPUT:-

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
    char stack[20],ip[20],opt[10][10][1],ter[10];
    int i,j,k,n,top=0,row,col;
    int len;
    for(i=0;i<10;i++)
    {
        stack[i]=NULL;ip[i]=NULL;
        for(j=0;j<10;j++)
        {
            opt[i][j][1]=NULL;
        }
    }
    printf("Enter the no.of terminals:");
    scanf("%d",&n);
    printf("\nEnter the terminals:");
    scanf("%s",ter);
    printf("\nEnter the table values:\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            printf("Enter the value for %c %c:",ter[i],ter[j]);
            scanf("%s",opt[i][j]);
        }

   }
    printf("\nOPERATOR PRECEDENCE TABLE:\n");
```

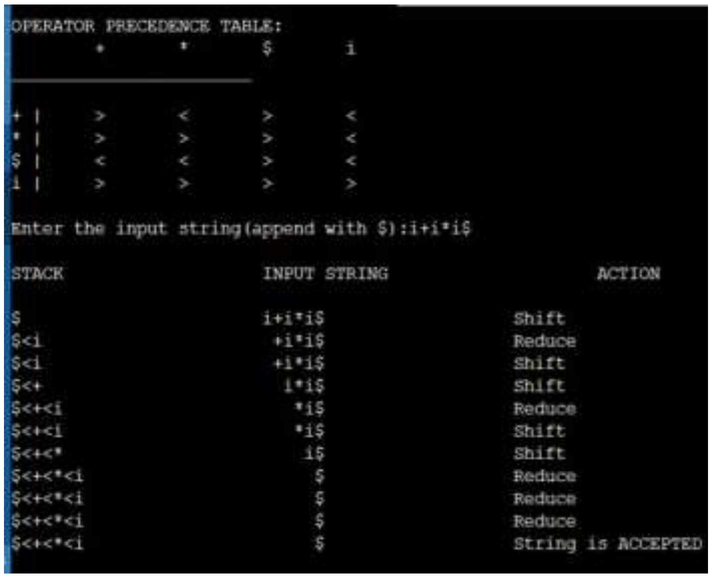
```
for(i=0;i<n;i++)
{
   printf("\t%c",ter[i]);
}
printf("\n_______________________");
printf("\n");
for(i=0;i<n;i++)
{
   printf("\n%c |",ter[i]);
   for(j=0;j<n;j++)
   {
      printf("\t%c",opt[i][j][0]);
   }
}
stack[top]='$';
printf("\n\nEnter the input string(append with $):");
scanf("%s",ip); i=0;
printf("\nSTACK\t\t\tINPUT STRING\t\t\tACTION\n");
printf("\n%s\t\t\t%s\t\t\t",stack,ip);
len=strlen(ip);
while(i<=len)
{
   for(k=0;k<n;k++)
   {
      if(stack[top]==ter[k])
      row=k;
      if(ip[i]==ter[k])
      col=k;
   }
   if((stack[top]=='$')&&(ip[i]=='$'))
   {
      printf("String is ACCEPTED");
      break;
   }
   else if((opt[row][col][0]=='<') ||(opt[row][col][0]=='='))
   {
      stack[++top]=opt[row][col][0];


      stack[++top]=ip[i];
      ip[i]=' ';
```

```
            printf("Shift %c",ip[i]); i++;
            }
            else
            {
                if(opt[row][col][0]=='>')
                {
                    while(stack[top]!='<')
                    {
                        --top;
                    }
                    top=top-1;
                    printf("Reduce");
                }
                else
                {
                    printf("\nString is not accepted");
                    break;
                }
            }
            printf("\n");
            printf("%s\t\t\t%s\t\t\t",stack,ip);
    }
    getch();
}
```

OUTPUT:-



```
Enter the no.of terminals:4

Enter the terminals:+*$i

Enter the table values:
Enter the value for + +:>
Enter the value for + *:<
Enter the value for + $:>
Enter the value for + i:<
Enter the value for * +:>
Enter the value for * *:>
Enter the value for * $:>
Enter the value for * i:<
Enter the value for $ +:<
Enter the value for $ *:<
Enter the value for $ $:>
Enter the value for $ i:<
Enter the value for i +:>
Enter the value for i *:>
Enter the value for i $:>
Enter the value for i i:>
```



```
OPERATOR PRECEDENCE TABLE:
        +       *       $       i
______________________________

+ |     >       <       >       <
* |     >       >       >       <
$ |     <       <       >       <
i |     >       >       >       >

Enter the input string(append with $):i+i*i$

STACK                   INPUT STRING                    ACTION

$                       i+i*i$                  Shift
$<i                      +i*i$                  Reduce
$<i                      +i*i$                  Shift
$<+                       i*i$                  Shift
$<+<i                      *i$                  Reduce
$<+<i                      *i$                  Shift
$<+<*                       i$                  Shift
$<+<*<i                      $                  Reduce
$<+<*<i                      $                  Reduce
$<+<*<i                      $                  Reduce
$<+<*<i                      $                  String is ACCEPTED
```