AIM:

Write a program to implement LEX in C.

SOURCECODE:

```
#include<stdio.h>
#include<ctype.h>
#include<conio.h>
FILE *fp,*fp1;
void main()
{
  char s[10];
  clrscr();
  fp=fopen("i.dat","r");
  fp1=fopen("o.dat","w");
  while(!feof(fp))
  {
    fscanf(fp,"%s",&s);
    check(s);
  }
}
check(char s[10])
{
  fp1=fopen("o.dat","a+");
  if(strcmp(s,"read")==0||strcmp(s,"write")==0||
    strcmp(s,"while")==0||strcmp(s,"for")==0||
    strcmp(s,"if")==0||strcmp(s,"else")==0||
```

```
    strcmp(s,"endif")==0||strcmp(s,"then")==0)
    fprintf(fp1,"%s->keyword \n",s);
  else
  if(!isalpha(s[0]))
  {
    if(strcmp(s,",")==0)
        fprintf(fp1,",->comma\n");
    if(strcmp(s,"(")==0)
        fprintf(fp1,"(->openbrace\n");
    if(strcmp(s,")")==0)
        fprintf(fp1,")->closebrace\n");
    if(strcmp(s,";")==0)
        fprintf(fp1,";->semicolon\n");
    if(strcmp(s,">")==0)
        fprintf(fp1,">->greaterthan->\n");
    if(strcmp(s,"<")==0)
        fprintf(fp1,"<->lessthan\n");
  }
  else
    fprintf(fp1,"%s->identifier\n",s);
    fclose(fp1);
}
```

OUTPUT:

Input:

Type i.dat

Read a,b

If(a>b)

Write a

Else

Write b

Output:

Type o.dat

Read keyword

a,b identifier

if(a>b) identifier

write keyword

a identifier

else keyword

write keyword

b identifier

b identifier

AIM:

Write a program to determine whether a given string is identifier or not.

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i=0,p;
  char a[10];
  clrscr();
  printf("enter a string:");
  scanf("%s",&a);
  for(i=0;a[i]!='\0';i++)
  {
    if((a[i]>='a'&&a[i]<='z')||
        (a[i]>='A'&&a[i]<='Z')||
        (a[i]>='0'&&a[i]<='9')||
        (a[i]=='_'))
    {
      p=1;
      break;
    }
    else
    p=0;break;
  }
```

```
    if(p==1)
    printf("given string is an identifier:");
    else
    printf("given string is not an identifier");
    getch();
}
```

OUTPUT:

Enter a string: pavan

Given string is an identifier

Enter a string:*&psbn

Given string is not an identifier

AIM:

Write a program to find whether a given string is constant or not.

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i,a;
  char s[10];
  clrscr();
  printf("enter a string:");
  scanf("%s",&s);
  for(i=0;s[i]!='\0'&&s[i]!='_';i++)
  {
    if((s[i]>='A'&&s[i]<='F')||
        (s[i]>='0'&&s[i]<='9')||
        (s[i]=='-')||(s[i]=='+'))
    {
        a=1;
        break;
    }
    else
        a=0;
        break;
  }
```

```
    if(a==1)
       printf("given string is a constant");
    else
       printf("given string is not a constant");
    getch();
}
```

OUTPUT:

Enter a string: 547

Given string is a constant.

Enter a string: panama

Given string is not a constant.

AIM:

Write a program to find whether a given string is relational operator or not.

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i,s=0;
  char str[10];
  clrscr();
  printf("enter a string:");
  scanf("%s",&str);
  for(i=0;str[i]!='\0';i++)
  {
   switch(s)
   {
    if((str[i]=='<')||(str[i]=='>')||(str[i]=='=')||(str[i]=='!'))
        case 0:
             if((str[i]=='<')||(str[i]=='>'))
             s=1;

            else
             if(str[i]=='=')
             s=3;
            else
```

```
            if(str[i]=='!')
            s=5;
           else
            s=99;
            break;

case 1:
        if(str[i]=='=')
          s=2;
        else
          s=99;
        break;
case 2:
        s=99;
        break;
case 3:
        if(str[i]=='=')
        s=4;
        else
        s=99;
        break;
case 4:
        s=99;
        break;
case 5:
```

```
            if(str[i]=='=')
            s=6;
            else
            s=99;
            break;
     case 6:
            s=99;
            break;
     }
    }
    if((s==1)||(s==4)||(s==6)||(s==2))
      printf("\n it is relational operator");
    else
      printf("\n not relational operator");
    getch();
}
```

OUTPUT:

Enter a string: pavan

Not a relational operator.

Enter a string: =

It is a relational operator.

AIM:

Write a program to find whether a given string is a comment or not.

SOURCECDE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i,f=0,n;
  char a[10];
  clrscr();
  printf("enter a string:");
  scanf("%s",&a);
  n=strlen(a);
  for(i=0;i<n;i++)
  {
    if((a[i]=='/')&&(a[i+1]=='*')&&(a[n-2]=='*')&&(a[n-1]=='/')||
        (a[i]=='/')&&(a[i+1]=='/'))
    {
        f=1;
        break;
    }
    else
        f=0;
  }
```

```
if(f==1)
      printf("the given string is comment");
   else
      printf("the given string is not comment");
   getch();
}
```

OUTPUT:

Enter a string: cse

The given string is not a comment.

Enter a string:/*dr.k.v.s.r.*/

The given string is a comment.

AIM:

Write a program to find even number of conjugative ones.

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i,j,sum=0;
  char msg[20];   clrscr();
  printf("\n enter the message stream:");
  scanf("%s",&msg);
  for(i=0;msg[i]!='\0';i++)
  {
    if(msg[i]==msg[i+1])
    {
        if(msg[i]=='1')
          sum++;
    }
  }
 if(sum%2==0)
    printf("even number of conjugative ones are%d\t",sum);
  else
    printf("there is no even number of conjugative ones");
  getch();
}
```

OUTPUT:

Enter the message stream: 11001111

Even number of conjugative ones are 4

AIM:

Write a program to find number odd number of conjugative ones in a given message stream.

SOURCECODE:

```
#include<stdio.h>

#include<conio.h>

void main()

{

  int i,j,sum=0;

  char msg[20];

  clrscr();

  printf("\n enter the message stream:");

  scanf("%s",&msg);

  for(i=0;msg[i]!='\0';i++)

  {

    if(msg[i]==msg[i+1])

    {

        if(msg[i]=='1')

        sum++;

    }

  }
```

```
if(sum%2!=0)
    printf("\n the string is having the odd number of conjugative ones %d\t",sum);
  else
    printf("\n the string is not having the odd number of conjugative ones");
  getch();
}
```

OUTPUT:

Enter the message stream: 111111000

The string is having the odd number of conjugative ones 5

Enter the message stream:11001100

The string is not having the odd number of conjugative ones.

\

AIM:

Write a program to find even number of ones in the given message stream.

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
main()
{
  int i,j,sum=0;
  char msg[20];
  clrscr();
  printf("enter the messeage stream:");
  scanf("%s",&msg);
  for(i=0;msg[i]!='\0';i++)
  {
    if(msg[i]=='1')
    sum++;
  }
  if(sum%2==0)
    printf("the string is having even number of ones%d",sum);
  else
    printf("the string is not having even number of ones");
  getch();
}
```

OUTPUT:

Enter the message stream:110011110

The string is having even number of ones 6

Enter the message stream: 111000

The string not having even number of ones.

AIM:

Write a program to find even number of ones in the given message stream.

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
main()
{
  int i,j,sum=0;
  char msg[20];
  clrscr();
  printf("enter the messeage stream:");
  scanf("%s",&msg);
  for(i=0;msg[i]!='\0';i++)
  {
    if(msg[i]=='1')
    sum++;
  }
  if(sum%2!=0)
    printf("the string is having odd number of ones%d",sum);
  else
    printf("the string is not having odd number of ones");
  getch();
}
```

OUTPUT:

Enter the message stream: 111000

The string is having odd number of ones 3

Enter the message stream:110011110

The string not having odd number of ones.

AIM:

Write a program to find whether a given string is even palindrome or not.

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
  int i,j,l,f;
  char s[100];
  clrscr();
  printf("enter the string:");
  gets(s);
  l=strlen(s);
  for(i=0,j=l-1;i<l;i++,j--)
  {
    if(s[i]!=s[j])
    {
        f=1;
        break;
    }
    else
        f=0;
  }
  if(f==0)
```

```
  {
    if(l%2==0)
        printf("\n it is an even palindrome");
    else
        printf("\n it is not an even palindrome");
  }
  else
    printf("string is not palindrome");
  getch();
}
```

OUTPUT:

Enter the string: pap

It is not an even palindrome.

Enter the string: paap

It is an even palindrome.

AIM:

Write a program to find whether a given string is odd palindrome or not.

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
  int i,j,l,f;
  char s[100];
  clrscr();
  printf("enter the string:");
  gets(s);
  l=strlen(s);
  for(i=0,j=l-1;i<l;i++,j--)
  {
    if(s[i]!=s[j])
    {
        f=1;
        break;
    }
    else
        f=0;
  }
  if(f==0)
```

```
  {
    if(l%2!=0)
        printf("\n it is an even palindrome");
    else
        printf("\n it is not an even palindrome");
  }
  else
    printf("string is not palindrome");
  getch();
}
```

OUTPUT:

Enter the string: apa

It is an odd palindrome.

Enter the string: paap

It is not an odd palindrome.

AIM:

Write a program for recursive descent parsing for a given input string.

SOURCECODE:

```
#include<stdio.h>
#include<ctype.h>
char str[15];
int p,chk=1;
void main()
{
  clrscr();
  printf("Enter the input string:");
  scanf("%s",&str);
  printf("%s",str);
  p=0;
  E();
  if(p==strlen(str)&&chk)
    printf("\nString is acceptable");
  else
  {
    printf("\nThe given input string:%s",str);
    printf("\nThe string is not acceptable");
  }
  getch();
}
E()
```

```
{
   printf("\nE->TE'");
   T();
   EPRIME();
}
EPRIME()
{
   if(str[p]=='+')
   {
      printf("\nE'->+TE'");
      p++;
      T();
      EPRIME();
   }
   else
      printf("\nE'->#");
}
T()
{
   printf("\nT->FT'");
   F();
   TPRIME();
}
TPRIME()
{
```

```
  if(str[p]=='*')
  {
    printf("\nT'->*FT'");
    p++;
    F();
    TPRIME();
  }
  else
    printf("\nT'->#");
}
F()
{
  if(isalpha(str[p]))
  {
    p++;
    printf("\nF->i");
  }
  else
  if(str[p]=='(')
  {
    printf("\nF->(E)");
    p++;
    E();
```

```
 if(str[p]==')')
            p++;
       else
            chk=1;
    }
    else
       chk=1;
}
```

OUTPUT:

Enter the input string:i+i*i

i+i*i

E-->TE’

T-->FT’

F-->i

T’-->#

E’-->+TE’

T-->FT’

F-->i

T’-->*FT’

F-->i

T’-->#

E’-->#

String is acceptable

Enter the input string :PO

PO

E-->TE’

T-->FT’

F-->i

T’-->#

E’-->#

The given input string: PO

The string is not acceptable.

AIM:

Write a program to implement LL(1) parsing table for a given input string

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
char nt[10]={'E','S','A','B'};
char t[10]={'a','b','#','$'};
int tc=4,ntc=5;
term(char);
nonterm(char);
main()
{
   char  pr[10][10]={"E->S#","E->S#","S->aB","S->bA","A->aS","A->bAA","B->bS",
                              "B->aBB"};
  char out[10][10][10]={' '};
  char ft[10][10]={"a","b","a","b","a","b","b","a"};
  int i,j,k,np=10,x,y,z,l,p;
  clrscr();
  for(k=0;k<np;k++)
  {
    l=nonterm(pr[k][0]);
    for(i=0;ft[k][i]!='\0';i++)
    {
      if(ft[k][i]=='&')
      {
```

```
        for(j=0;ft[l-i][j]!='\0';j++)
        {
          p=term(ft[l-i][j]);
          strcpy(out[l][p],pr[k]);
        }
    }
    else
    {
        p=term(ft[k][i]);
        strcpy(out[l][p],pr[k]);
    }
  }
}
 for(i=0;i<tc;i++)
   out[0][i+1][0]=t[i];
 for(i=0;i<ntc;i++)
   out[i+1][0][0]=nt[i];
 printf("\n\t\t\tLL(1) PARSING TABLE\n\n\n");
 for(i=0;i<ntc;i++)
 {
   printf("\n\n");
   for(j=0;j<=tc;j++)
     printf("%s\t",out[i][j]);
 }
getch();
```

```
}
term(char c)
{
  int i;
  for(i=0;i<tc;i++)
    if(c==t[i])
      return i+1;
  return 0;
}
nonterm(char c)
{
  int i;
  for(i=0;i<ntc;i++)
    if(c==nt[i])
      return i+1;
  return 0;
}
```

OUTPUT:

LL(1) PARSING TABEL

	a	b	#	$
E	E□S#	E□S#		
S	S□aB	S□bA		
A	A□aS	A□bAA		
B	B□aBB	B□bS		

AIM:

Write a program to convert infix expression to postfix expression.

SOURCECODE:

```
#include<stdio.h>
#include<string.h>
#define operand(x) (x>='a'&&x<='z'||x>='A'&&x<='Z'||x>='0'&&x<='9')
char infix[30],postfix[30],stack[30];
int top,i=0;
void init()
{
   top=-1;
}
void push(char x)
{
   stack[++top]=x;
}
char pop()
{
   return(stack[top--]);
}
int isp(char x)
{
   int y;
   y=(x=='('?0:x=='^'?4:x=='*'?2:x=='/'?2:x=='+'?1:x=='-'?1:x==')'?6:-1);
   return y;
```

```
}
int icp(char x)
{
   int y;
   y=(x=='('?4:x=='^'?4:x=='*'?2:x=='/'?2:x=='+'?1:x=='-'?1:x==')'?6:-1);
   return y;
}
void infixtopostfix()
{
   int j,l=0;
   char x,y;
   stack[++top]='\0';
   for(j=0;(x=infix[i++])!='\0';j--)
      if(operand(x))
            postfix[l++]=x;
      else
      if(x==')')
            while((y=pop())!='('
               postfix[l++]=y;
      else
      {
            while(isp(stack[top])>=icp(x))
            postfix[l++]=pop();
            push(x);
      }
```

```
    while (top>=0)
    postfix[l++]=pop();
}
void main()
{
  init();
  clrscr();
  printf("Enter an infix expression :\n");
  scanf("%s",infix);
  infixtopostfix();
  printf("The resulting postfix expression is %s",postfix);
  getch();
}
```

OUTPUT:

Enter an infix expression: (a+b)*(c+d)

The resulting postfix expression is: ab+cd+*

AIM:

Write a program to prepare TAC for the given postfix notation.

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
main()
{
  char op[10][3],arg1[10][3],arg2[10][3],result[10][3],s[18],t[3],
       stack[15][3],gen[9][3]={"T1","T2","T3","T4","T5","T6","T7","T8","T9"};
  int i,j,k=0,p=0;
  clrscr();
  printf("Enter the string in postfix notation\n");
  fflush(stdin);
  scanf("%s",&s);
  printf("Given %s\n",s);
  for(i=0;i<strlen(s);i++)
  {
    t[0]=s[i];
    t[1]='\0';
    if(isalnum(s[i]))
        strcpy(stack[k++],t);
    else
```

```
      {
            strcpy(op[p],t);
            strcpy(arg2[p],stack[k-1]);
            strcpy(arg1[p],stack[k-2]);
            strcpy(result[p],gen[p]);
            strcpy(stack[k-2],result[p]);
            p++,k--;
      }
   }
   printf("Required TAC code is:");
   for(i=0;i<p;i++)
      printf("\n%s:=%s%s%s",result[i],arg1[i],op[i],arg2[i]);
   getch();
}
```

OUTPUT:

Enter the string in post notation:

ab+cd+*

Required TAC code is:

T1:=a+b

T2:=c+d

T3:=T1*T2

AIM:

Write a program for code optimization.

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
  int found,l;
  char s[50];
  FILE *f1,*f2;
  clrscr();
  f1=fopen("i.dat","r");
  f2=fopen("o.dat","w");
  while(!feof(f1))
  {
    found=0;
    fscanf(f1,"%s",s);
    l=strlen(s);
    if((s[l-1]=='1'&&s[l-2]=='*')||(s[l-1]=='0'&&s[l-2]=='+'))
        found=1;
    if(found==0)
        fprintf(f2,"%s\n",s);
  }
```

```
    fclose(f1);
    fclose(f2);
    getch();
}
```

OUTPUT:

Input:

i.dat

```
t1=t1*1
t2=t2+0
t3=t1+t2
t4=t3*c
t5=t4+2
a=t5
```

OUTPUT:

o.dat

```
t1=t1+t2
t4=t3*c
t5=t4+2
a=t5
```

AIM:

Write a program for heap storage allocation.

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
void main()
{
  int t,mcap,np,temp,a,i,j,k,q[50],pv[50],pos[50],s,posval[10],min;
  clrscr();
  printf("Enter memory capacity:");
  scanf("%d",&mcap);
  printf("Enter the no. of partitions:");
  scanf("%d",&np);
  printf("Enter the partition values:");
  for(i=0;i<np;i++)
    scanf("%d",&pv[i]);
  printf("Enter the values in queue:");
  for(i=0;i<5;i++)
    scanf("%d",&q[i]);
  for(j=0;j<5;j++)
  {
    k=0;
    for(i=0;i<np;i++)
    {
```

```
            temp=pv[i]-q[j];
            if(temp>0)
            {
               pos[k]=i;
               posval[k]=pv[i];
               k++;
            }
        }
        min=posval[0];
        for(a=0;a<k;a++)
        {
            if(posval[a]<min)
            min=posval[a];
        }
        printf("\n%dkb of memory is allocated to memory fixed partition value %d",min);
    }
    getch();
}
```

OUTPUT:

Enter memory capacity: 220

Enter the number of partitions: 4

Enter partition values: 25 42 65 88

Enter the values in queue: 24 75 48 32

24kb of memory is allocated to memory fixed partition value 25

75 kb of memory is allocated to memory fixed partition value 88

48 kb of memory is allocated to memory fixed partition value 65

32 kb of memory is allocated to memory fixed partition value 42

AIM:

Write a program to check whether a given grammar is ambiguous or not.

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
#include<string.h>
char a[60];
int i=0,s=0;
int amb();
void main()
{
  int i,flag=1;
  char p[50];
  clrscr();
  printf("Enter the grammar:\n");
  gets(p);
  for(i=0;p[i]!='\0';i++)
  {
    if((p[0]==p[i+2])&&(p[0]==p[i+6]))
    {
        if(p[i]=='1')
        {
          flag=1;
          break;
```

```
        }
        else
        if(p[i]=='+'||p[i]=='-'||p[i]=='*'||p[i]==')'||p[i]=='/')
        {
          flag=2;
          break;
        }
        else
        {
          flag=0;
        }
    }
}
if(flag==0)
{
   amb();
   printf("\nThe given grammar is ambiguous");
}
else
if(flag==1)
{
   unamb();
   printf("\nThe given grammar is unambiguous");
}
else
```

```
  if(flag==2)
  {
    if(amb())
        printf("\nString is accepted");
    else
        printf("\nString is not accepted");
  }
  else
  {
    if(unamb())
        printf("String is accepted\n");
    else
        printf("String is not accepted\n");
  }
  getch();
}
amb()
{
  printf("Enter the string:");
  scanf("%s",&a);
  if(islower(a[i]))
  {
    i++;
    s++;
  }
```

```
else
if(a[i]=='(')
{
   i++;
   s++;
   if(a[i]==')')
   {
        i++;
        s++;
   }
   return 0;
}
else
if(a[i]=='*')
{
   i++;
   s++;
}
else
if(a[i]=='*'||a[i]=='-'||a[i]=='/')
{
   i++;
   s++;
}
s++;
```

```
  if(s>1)
     printf("Generates more than one parse tree");
}
unamb()
{
  printf("Enter the string");
  scanf("%s",&a);
  if(islower(a[i]))
     i++;
  else
  if(a[i]=='(')
  {
     i++;
     if(a[i]==')')
          i++;
     else
          s++;
  }
  if(a[i]=='+'||a[i]=='-'||a[i]=='(')
  {
     i++;
     s++;
  }
  if(a[i]=='*')
  {
```

```
    i++;
    s++;
  }

if(s==1)
    printf("Generates one parse tree");
}
```

OUTPUT:

Enter the grammar:

E->E+E E->i

Enter the string:i+i*i

Generates more than one parse tree

String is accepted

Enter the grammar:

E->E E->i

Enter the string: i

The given grammar is unambiguous.

AIM:

Write a program to implement SHIFT REDUCE PARSER for a given grammar.

SOURCECODE:

```
#include<stdio.h>
int a[10]={121,131,514,6,181};
int stack[20],top=-1;
void push(int item)
{
  if(top>=20)
  {
    printf("overflow");
    exit(1);
  }
  top++;
  stack[top]=item;
}
char convert(int item)
{
  char ch;
  switch(item)
  {
    case 1:ch='e';break;
    case 2:ch='*';break;
    case 3:ch='+';break;
    case 4:ch='(';break;
```

```
    case 5:ch=')';break;
    case 6:ch='i';break;
    case 7:ch='$';break;
    case 8:ch='=';break;
  }
  return(ch);
}
int action(int item)
{
  int i;
  for(i=0;i<=4;i++)
  {
    if(a[i]==item)
      return 1;
  }
  return -1;
}
void main()
{
  char ipstr[20];
  int ip[20],i,s,sum,j,l,m,n,cnt;
  clrscr();
  printf("\nEnter the input string:");
  scanf("%s",ipstr);
  for(i=0;ipstr[i]!='\0';i++)
```

```
{
  switch(ipstr[i])
  {
    case 'e':s=1;break;
    case '*':s=2;break;
    case '+':s=3;break;
    case '(':s=4;break;
    case ')':s=5;break;
    case 'i':s=6;break;
    case '$':s=7;break;
    case '=':s=8;break;
    default:printf("error");
            exit(1);
  }
  ip[i]=s;
}
ip[i]=-1;
i=0;
push(7);
printf("\t stack \t input \t  action \n");
printf("\t----------------------------\n\n");
while(1)
{
  printf("\t");
  for(m=0;m<=top;m++)
```

```
 printf("%c",convert(stack[m]));
 printf("\t");
for(n=i;ip[n]!=-1;n++)
 printf("%c",convert(ip[n]));
 printf("\t");
 sum=0;
 if((stack[top]==1)&&(stack[top-1]==7)&&(ip[i]==7))
 {
   printf("\taccept");
   exit(1);
 }
 else
 {
   l=1;
   while(l<=top)
   {
   j=l;
   sum=0;cnt=0;
   for(;j<=top;j++)
   {
        sum=sum*10+stack[j];
        cnt++;
   }
   s=action(sum);
   if(s==1)
```

```
      {
          top=top-cnt;
          push(1);
          printf("\treduce");
          break;
      }
      l++;
      }
      if(s==-1&&ip[i]==7)
      {
      printf("\terror");
      exit(1);
      }
    }
   if(ip[i]!=7)
    {
      push(ip[i]);
      i++;
      printf("\tshift");
    }
    printf("\n");
  }
  getch();
}
```

OUTPUT:

Enter the input string:i*i+i$

Stack	input	action
$	i*i+i$	shift
$i	*i+i$	reduce shift
$e*	i+i$	shift
$e*i	+i$	reduce shift
$e*e+	i$	shift
$e*e+i	$	reduce
$e*e+e	$	reduce
$e*e	$	reduce
$e	$	accept

Enter the input string:i+i$

Stack	input	action
$	i+i$	shift
$i	+i$	reduce shift
$e+	i$	shift
$e+i	$	reduce
$e+e	$	reduce
$e	$	accept