Program 7:
AIM: To Implement SLR(1) Parsing algorithm

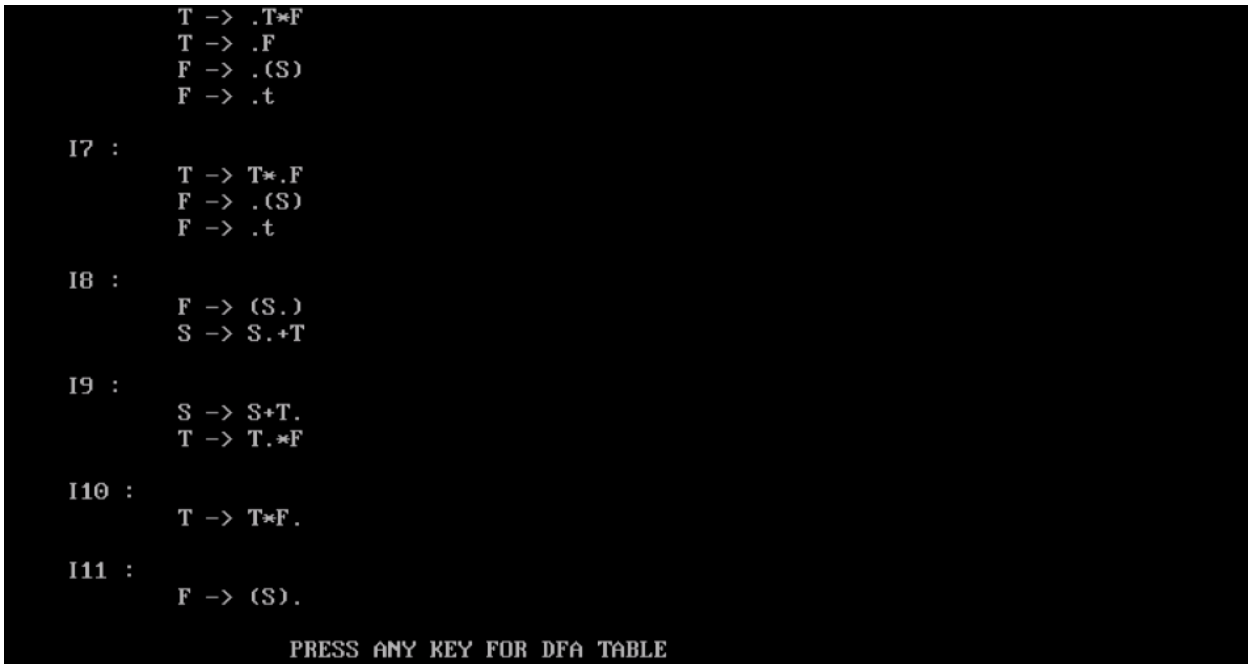
```
#include<stdio.h>
#include<string.h>
int axn[][6][2]={
{{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},
{{-1,-1},{100,6},{-1,-1},{-1,-1},{-1,-1},{102,102}},
{{-1,-1},{101,2},{100,7},{-1,-1},{101,2},{101,2}},
{{-1,-1},{101,4},{101,4},{-1,-1},{101,4},{101,4}},
{{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},
{{-1,-1},{101,6},{101,6},{-1,-1},{101,6},{101,6}},
{{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},
{{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},
{{-1,-1},{100,6},{-1,-1},{-1,-1},{100,1},{-1,-1}},
{{-1,-1},{101,1},{100,7},{-1,-1},{101,1},{101,1}},
{{-1,-1},{101,3},{101,3},{-1,-1},{101,3},{101,3}},
{{-1,-1},{101,5},{101,5},{-1,-1},{101,5},{101,5}}
};//Axn Table
int gotot[12][3]={1,2,3,-1,-1,-1,-1,-1,-1,-1,-1,-1,8,2,3,-1,-1,-1,
-1,9,3,-1,-1,10,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1}; //GoTo table
int a[10];
char b[10];
int top=-1,btop=-1,i;
void push(int k)

{
if(top<9)
a[++top]=k;
}
void pushb(char k)
{
if(btop<9)
b[++btop]=k;
}
char TOS()
{
return a[top];
}
void pop()
{
if(top>=0)
top--;
}
void popb()
{
if(btop>=0)
b[btop--]='\0';
```

```
}
void display()
{
for(i=0;i<=top;i++)
printf("%d%c",a[i],b[i]);
}
void display1(char p[],int m) //Displays The Present Input String
{
int l;
printf("\t\t");
for(l=m;p[l]!='\0';l++)
printf("%c",p[l]);
printf("\n");
}
void error()
{
printf("Syntax Error");
}
void reduce(int p)
{
int len,k,ad;
char src,*dest;
switch(p)
{

case 1:dest="E+T";
src='E';
break;
case 2:dest="T";
src='E';
break;
case 3:dest="T*F";
src='T';
break;
case 4:dest="F";
src='T';
break;
case 5:dest="(E)";
src='F';
break;
case 6:dest="i";
src='F';
break;
default:dest="\0";
src='\0';
break;
}
for(k=0;k<strlen(dest);k++)
```

```
{
pop();
popb();
}
pushb(src);
switch(src)
{
case 'E':ad=0;
break;
case 'T':ad=1;
break;
case 'F':ad=2;
break;
default: ad=-1;
break;
}
push(gotot[TOS()][ad]);
}
int main()
{
int j,st,ic;
char ip[20]="\0",an;
// clrscr();
printf("Enter any String\n");
scanf("%s",ip);
push(0);

display();
printf("\t%s\n",ip);
for(j=0;ip[j]!='\0';)
{
st=TOS();
an=ip[j];
if(an>='a'&&an<='z') ic=0;
else if(an=='+') ic=1;
else if(an=='*') ic=2;
else if(an=='(') ic=3;
else if(an==')') ic=4;
else if(an=='$') ic=5;
else {
error();
break;
}
if(axn[st][ic][0]==100)
{
pushb(an);
push(axn[st][ic][1]);
display();
```

```
j++;
display1(ip,j);
}
if(axn[st][ic][0]==101)
{
reduce(axn[st][ic][1]);
display();
display1(ip,j);
}
if(axn[st][ic][1]==102)
{
printf("Given String is accepted \n");
// getch();
break;
}
/* else
{
printf("Given String is rejected \n");
break;
}*/
}
return 0;
}
```

Output:



Enter any String
a+a*a$
0 a+a*a$
0a5 +a*a$
0F3 +a*a$
0T2 +a*a$
0E1 +a*a$
0E1+6 a*a$
0E1+6a5 *a$
0E1+6F3 *a$
0E1+6T9 *a$
0E1+6T9*7 a$
0E1+6T9*7a5 $
0E1+6T9*7F10 $
0E1+6T9 $
0E1 $
Given String is accepted

Program 8:
AIM: To Design LALR bottom up parser for the given language.

```
<parser.l>
%{
#include<stdio.h>
#include "y.tab.h"
%}
%%
[0-9]+ {yylval.dval=atof(yytext);
return DIGIT;
}
\n|. return yytext[0];
%%
<parser.y>
%{
/*This YACC specification file generates the LALR parser for the program
considered in experiment 4.*/
#include<stdio.h>
%}
%union
{
double dval;
}
%token <dval> DIGIT
%type <dval> expr
%type <dval> term
%type <dval> factor
%%
line: expr '\n' {
printf("%g\n",$1);
}

;
expr: expr '+' term {$$=$1 + $3 ;}
| term
;
term: term '*' factor {$$=$1 * $3 ;}
| factor
;
factor: '(' expr ')' {$$=$2 ;}
| DIGIT
;
%%
int main()
{
yyparse();
```

```
}
yyerror(char *s)
{
printf("%s",s);
}
```

INPUT:

```
$lex parser.l
$yacc –d parser.y
$cc lex.yy.c y.tab.c –ll –lm
$./a.out
```

OUTPUT:

```
2+3
5.0000
```