# PREDICTIVE PARSING

NAME: ATHRESH KUMAR LABDE
RA1911033010146
M1

**Aim**: A program for Predictive Parsing.
**Algorithm**:
1. Start the program.
2. Initialize the required variables.
3. Get the number of coordinates and productions from the user.
4. Perform the following for (each production A → α in G) { for (each terminal a in FIRST(α)) add A → α to M[A, a]; if (ε is in FIRST(α)) for (each symbol b in FOLLOW(A)) add A → α to M[A, b];
5. Print the resulting stack.
6. Print if the grammar is accepted or not.
7. Exit the program.

**CODE:**
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char fin[10][20],st[10][20],ft[20][20],fol[20][20];
int a=0,e,i,t,b,c,n,k,l=0,j,s,m,p;
clrscr();
printf("enter the no. of nonterminals\n");
scanf("%d",&n);
printf("enter the productions in a grammar\n");
for(i=0;i<n;i++)
scanf("%s",st[i]);
for(i=0;i<n;i++)
fol[i][0]='\0';
for(s=0;s<n;s++)
{
for(i=0;i<n;i++)
{
j=3;
```

```c
l=0;
a=0;
l1:if(!((st[i][j]>64)&&(st[i][j]<91)))
{
for(m=0;m<l;m++)
{
if(ft[i][m]==st[i][j])
goto s1;
}
ft[i][l]=st[i][j];
l=l+1;
s1:j=j+1;
}
else
{
if(s>0)
{
while(st[i][j]!=st[a][0])
{
a++;
}
b=0;
while(ft[a][b]!='\0')
{
for(m=0;m<l;m++)
{
if(ft[i][m]==ft[a][b])
goto s2;
}
ft[i][l]=ft[a][b];
l=l+1;
s2:b=b+1;
}
}
}
while(st[i][j]!='\0')
{
if(st[i][j]=='|')
{
```

```c
j=j+1;
goto l1;
}
j=j+1;
}
ft[i][l]='\0';
}
}
printf("first \n");
for(i=0;i<n;i++)
printf("FIRS[%c]=%s\n",st[i][0],ft[i]);
fol[0][0]='$';
for(i=0;i<n;i++)
{
k=0;
j=3;
if(i==0)
l=1;
else
l=0;
k1:while((st[i][0]!=st[k][j])&&(k<n))
{
if(st[k][j]=='\0')
{
k++;
j=2;
}
j++;
}
j=j+1;
if(st[i][0]==st[k][j-1])
{
if((st[k][j]!='|')&&(st[k][j]!='\0'))
{
a=0;
if(!((st[k][j]>64)&&(st[k][j]<91)))
{
for(m=0;m<l;m++)
{
```

```c
if(fol[i][m]==st[k][j])
goto q3;
}
fol[i][l]=st[k][j];
l++;
q3:
}
else
{
while(st[k][j]!=st[a][0])
{
a++;
}
p=0;
while(ft[a][p]!='\0')
{
if(ft[a][p]!='@')
{
for(m=0;m<l;m++)
{
if(fol[i][m]==ft[a][p])
goto q2;
}
fol[i][l]=ft[a][p];
l=l+1;
}
else
e=1;
q2:p++;
}
if(e==1)
{
e=0;
goto a1;
}
}
}
else
{
```

```c
a1:c=0;
a=0;
while(st[k][0]!=st[a][0])
{
a++;
}
while((fol[a][c]!='\0')&&(st[a][0]!=st[i][0]))
{
for(m=0;m<l;m++)
{
if(fol[i][m]==fol[a][c])
goto q1;
}
fol[i][l]=fol[a][c];
l++;
q1:c++;
}
}
goto k1;
}
fol[i][l]='\0';
}
printf("follow \n");
for(i=0;i<n;i++)
printf("FOLLOW[%c]=%s\n",st[i][0],fol[i]);
printf("\n");
s=0;
for(i=0;i<n;i++)
{
j=3;
while(st[i][j]!='\0')
{
if((st[i][j-1]=='|')||(j==3))
{
for(p=0;p<=2;p++)
{
fin[s][p]=st[i][p];
}
t=j;
```

```c
for(p=3;((st[i][j]!='|')&&(st[i][j]!='\0'));p++)
{
fin[s][p]=st[i][j];
j++;
}
fin[s][p]='\0';
if(st[i][k]=='@')
{
b=0;
a=0;
while(st[a][0]!=st[i][0])
{
a++;
}
while(fol[a][b]!='\0')
{
printf("M[%c,%c]=%s\n",st[i][0],fol[a][b],fin[s]);
b++;
}
}
else if(!((st[i][t]>64)&&(st[i][t]<91)))
printf("M[%c,%c]=%s\n",st[i][0],st[i][t],fin[s]);
else
{
b=0;
a=0;
while(st[a][0]!=st[i][3])
{
a++;
}
while(ft[a][b]!='\0')
{
printf("M[%c,%c]=%s\n",st[i][0],ft[a][b],fin[s]);
b++;
}
}
s++;
}
if(st[i][j]=='|')
```

```
j++;
}
}
getch();
}
```

**Output:**

```
PROBLEMS  4    OUTPUT   DEBUG CONSOLE   TERMINAL

PS C:\Users\athre\Desktop\laabbb> cd "c:\Users\athre\Desktop\laabbb\" ; if ($?) { gcc parsingtable.c -o
E->TA
A->+TA|^
T->FB
B->*FB|^
F->t|(E)
FIRST OF E: ( t
FIRST OF A: + ^
FIRST OF T: ( t
FIRST OF B: * ^
FIRST OF F: ( t

FOLLOW OF E: $ * +
FOLLOW OF A: $ * +
FOLLOW OF T: $ * +
FOLLOW OF B: $ * +
FOLLOW OF F: $ * +

FIRST OF E->TA: ( t
FIRST OF A->+TA: +
FIRST OF A->^: ^
FIRST OF T->FB: ( t
FIRST OF B->*FB: *
FIRST OF B->^: ^
FIRST OF F->t: t
FIRST OF F->(E: (


        *************** LL(1) PARSING TABLE *******************
        ---------------------------------------------------------
            $         (         *         +         t
E                   E->TA                          E->TA
A         A->^                 A->^      A->^
T                   T->FB                          T->FB
B         B->^                 B->^      B->^
F                   F->(E                          F->t
PS C:\Users\athre\Desktop\laabbb> []
```

**Result :**

The program was successfully compiled and run.