Date
22/02/23

EXPERIMENT - 6

PREDICTIVE PARSING TABLE

DO:1

AIM: To construct a program to implement the predictive parsing table for the given grammar.

ALGORITHM:

1. Start the program.
2. Initialize the variables given.
3. Get the no of coordinates and productions from the user.
4. Perform the follow

for (each production $A \rightarrow \alpha$ in G)
for (each terminal A in First(α))
add $A \rightarrow \alpha$ to M[A, a];
if (ε is in FIRST(α))
for (each symbol b in follow(A))
add $A \rightarrow \alpha$. to M[A, b].

5. Print the results stack.
6. Print if the grammar is correct or not
7. End the program.

PROGRAM CODE:

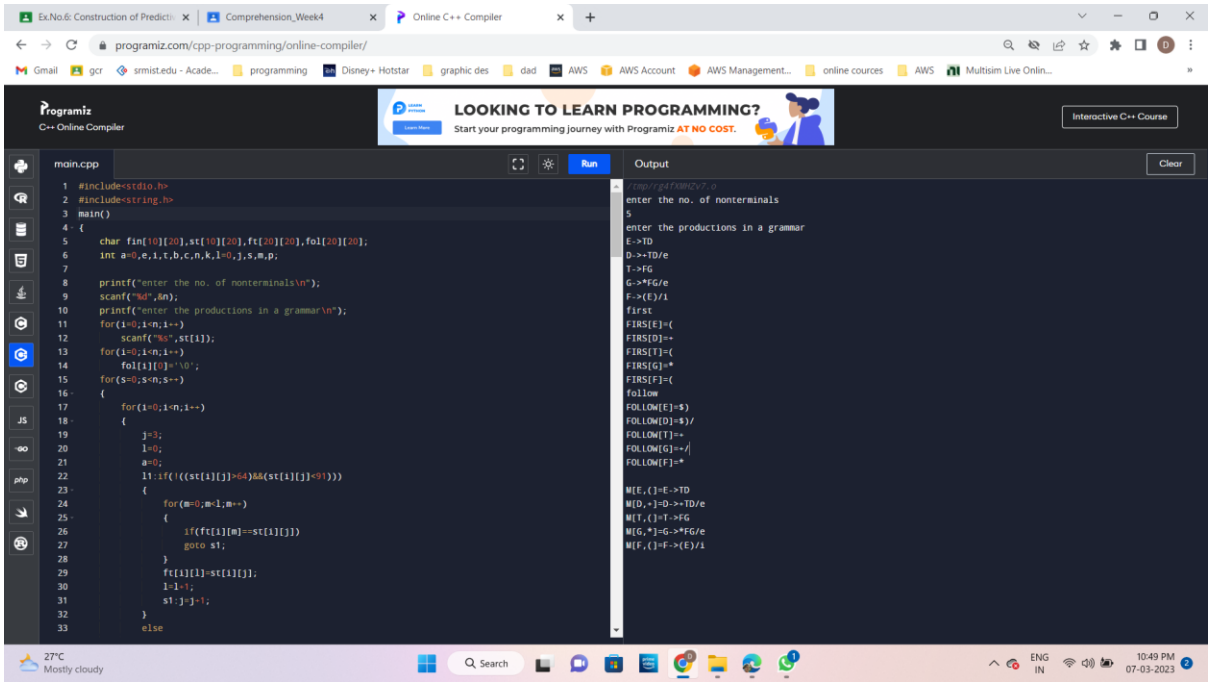
```
#include <stdio.h>
#include<string.h>
int main()
{ char fin[10][20], st[10][20], ft[20][20], fol[20][20];
int a=0,e,i,t,b,c,n,k,l=0,j,s,m,p;
printf("enter the no of productions\n");
```

```
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++)
      ft[i][0]='\0';
     for(s=0;s<n;s++)
     { for(i=0;i<n;i++)
     { j=3;
       l=0;
       a=0;
if: 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]=='1')
{  j = j+1;
   goto l1;}
 j = j+1; }
ft[i][l] = '\0' ; }
}
printf ("first pos \n");
for (i=0;i<n;i++)
printf ("FIRST [%d,%c] = %d \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]!='1')&&(st[k][j]!='\0'))
```

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

```
{ for (m=0; m<l; m++)
 { if (fol[i][m] == fol[0][c])
  goto q1; }
fol[i][l] = fol[a][c])
 goto q1; }
 fol[i][l] = fol[a][c];
 l++;
 q1: c++; } }
 goto k1; }
 fol[i][l] = '\0'; }
printf("follow pos\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;
for (p=3; (st[i][j] != '|') && (st[i][j] !=
{ fin[s][p] = st[i][j];
 j++; }
```


Ex.No.6: Construction of Predicti
Comprehension_Week4
Online C++ Compiler
programiz.com/cpp-programming/online-compiler/
Programiz
C++ Online Compiler
LOOKING TO LEARN PROGRAMMING?
Start your programming journey with Programiz AT NO COST.
Interactive C++ Course
main.cpp
Run
Output
Clear
#include<stdio.h>
#include<string.h>
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;
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;
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
enter the no. of nonterminals
5
enter the productions in a grammar
E->TD
D->+TD/e
T->FG
G->*FG/e
F->(E)/i
first
FIRS[E]=(
FIRS[D]=+
FIRS[T]=(
FIRS[G]=*
FIRS[F]=(
follow
FOLLOW[E]=$)
FOLLOW[D]=$)/
FOLLOW[T]=+
FOLLOW[G]=+/
FOLLOW[F]=*
M[E,(]=E->TD
M[D,+]=D->+TD/e
M[T,(]=T->FG
M[G,*]=G->*FG/e
M[F,(]=F->(E)/i
27°C
Mostly cloudy
Search
ENG
IN
10:49 PM
07-03-2023

Output:

enter no of productions

5

enter productions in a grammar

E → TD

D → +TD | e

T → FG

G → *FG | e

F → (E) | i

~~E → TD~~

first pos

FIRST [E] = (i

FIRST [D] = +e

FIRST [T] = (i

FIRST [G] = *e

FIRST [F] = (i

follow Pos

FOLLOW [E] = $)

FOLLOW [D] = $)

FOLLOW [T] = +$)

FOLLOW [G] = +$)

FOLLOW [F] = *+$)

M [F, ( ] = E → TD

M [E, i] = E → TD

M [D, +] = D → +TD

M [D, $] = D → e

M [D,)] = D → e

M [T, ( ] = T → FG

M [T, i] = T → FG

M [G, *] = G → *FG

M [G, +] = G → e

o/p verified
Rathi
22/2/23

```
fin[s][p] = '\0';
if(st[i][t] == 'e')
{ 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++; }}
 return 0; }
```

RESULT: Hence predictive parsing table is constructed for given grammar.