Rahul Goel

RA1911030010094

DATE: 3/3/22

COMPILER DESIGN EXP – 6

PREDICTIVE PARSING TABLE

AIM:

To construct a predictive parsing table for a given grammar.

ALGORITHM:

- Input the no of productions in the grammar
- Input the productions from the user in the form of a string
- Eliminate any left recursion or left factoring if present in the grammar - compute the first for the given grammar using the following rules: 1. If x is a terminal, then FIRST(x) = { ‘x’ }
    2. If x-> Є, is a production rule, then add Є to FIRST(x).
    3. If X->Y1 Y2 Y3….Yn is a production, FIRST(X) = FIRST(Y1)
    4. If FIRST(Y1) contains Є then FIRST(X) = { FIRST(Y1) – Є } U { FIRST(Y2) }
    5. If FIRST (Yi) contains Є for all i = 1 to n, then add Є to FIRST(X)
- Compute the follow for the given grammar using the following rules: 1. FOLLOW(S) = { $ } // where S is the starting Non-Terminal
    2. If A -> pBq is a production, where p, B and q are any grammar symbols, then everything in FIRST(q) except Є is in FOLLOW(B)
    3. If A->pB is a production, then everything in FOLLOW(A) is in FOLLOW(B). 4. If A->pBq is a production and FIRST(q) contains Є, then FOLLOW(B) contains { FIRST(q) – Є } U FOLLOW(A)
- Print the first and follow productions derived
- Print the predictive parsing table using the first and follow produced

CODE:

```
#include <iostream>

#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++)
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
{
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]=='|')
{
j=j+1;
goto l1;
}
j=j+1;
}
ft[i][l]='\0';
}
}
printf("first productions\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++)

{

if(fol[i][m]==st[k][j])

goto q3;

}
fol[i][l]=st[k][j];

l++;

q3:p++;

}

else

{
```

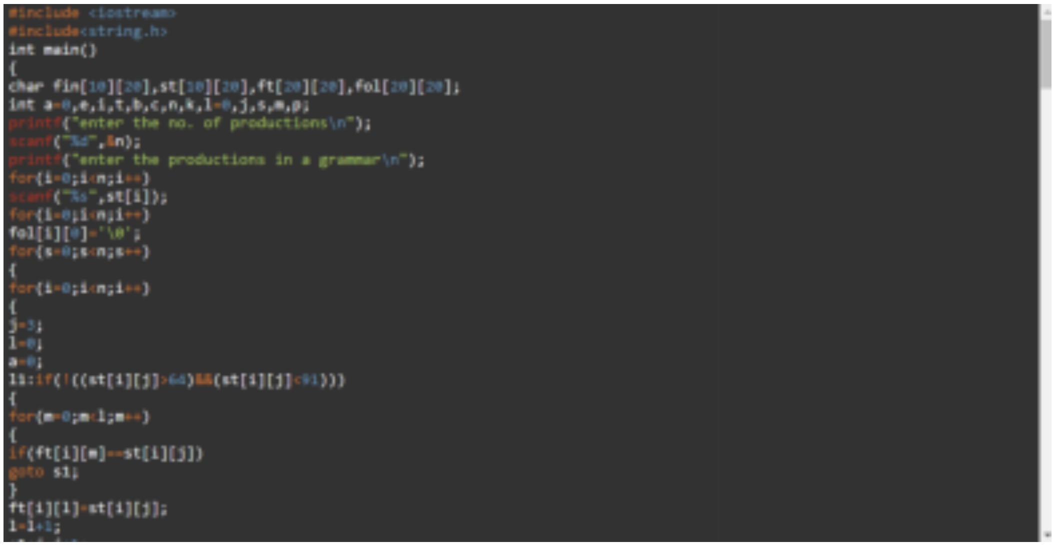
```
while(st[k][j]!=st[a][0])

{

a++;

}

p=0;

while(ft[a][p]!='\0')

{

if(ft[a][p]!='e')

{

for(m=0;m<l;m++)

{

if(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

{

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 productions\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++)
{
```

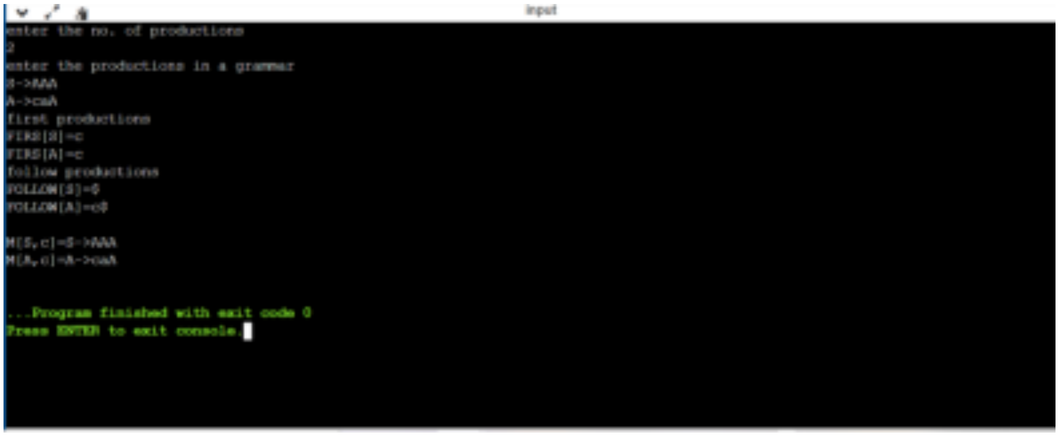
```
fin[s][p]=st[i][p];
}
t=j;
for(p=3;((st[i][j]!='|')&&(st[i][j]!='\0'));p++) {
fin[s][p]=st[i][j];
j++;
}
fin[s][p]='\0';
if(st[i][t]=='e')
{
b=0;
a=0;
while(st[a][0]!=st[i][0])
{
a++;
}
while(fol[a][b]!='\0')
{
printf("M[%c,%c]=%s\n",st[i][0],fol[a][b],fin[s]);
b++;
}
}
else if(!((st[i][t]>64)&&(st[i][t]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

}
```

OUTPUT SCREENSHOTS:



```
#include <iostream>
#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++)
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;
```



```
enter the no. of productions
2
enter the productions in a grammar
S->AAA
A->caA
first productions
FIRS[S]=c
FIRS[A]=c
follow productions
FOLLOW[S]=$
FOLLOW[A]=c$

M[S,c]=S->AAA
M[A,c]=A->caA


...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

The predictive parsing table for the given grammar was constructed successfully.