EXP 5

FIRST AND FOLLOW

AIM: To write a program for displaying the first and follow of a given production

ALGORITHM:

For computing the first:

1. If X is a terminal then FIRST(X) = {X}

Example: F -> (E) | id

We can write it as FIRST(F) -> { (, id }

2. If X is a non terminal like E -> T then to get

FIRST(E) substitute T with other productions until you get a terminal as the first symbol

3. If X -> ε then add ε to FIRST(X).

For computing the follow:

1. Always check the right side of the productions for a non-terminal, whose FOLLOW set is being found. (never see the left side).

2. (a) If that non-terminal (S,A,B…) is followed by any terminal (a,b…,*,+,(,)…) , then add that “terminal” into FOLLOW set.

(b) If that non-terminal is followed by any other non-terminal then add “FIRST of other nonterminal” into FOLLOW set.

CODE:

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
#define max 20
char prod[max][10];
char ter[10],nt[10];
char first[10][10],follow[10][10];
int eps[10];
int count=0;
int findpos(char ch)
{
int n;
for(n=0;nt[n]!='\0';n++)
if(nt[n]==ch)
break;
if(nt[n]=='\0')
return 1;
return n;
}

int IsCap(char c)
{
if(c >= 'A' && c<= 'Z')
return 1;
return 0;
}
```

```
void add(char *arr,char c)
{
int i,flag=0;
for(i=0;arr[i]!='\0';i++)
{
if(arr[i] == c)
{
flag=1;
break;
}
}
if(flag!=1)
arr[strlen(arr)] = c;
}
void addarr(char *s1,char *s2)
{
int i,j,flag=99;
for(i=0;s2[i]!='\0';i++)
{
flag=0;
for(j=0;;j++)
{
if(s2[i]==s1[j])
{
flag=1;
break;
}
if(j==strlen(s1) && flag!=1)
{
```

```
s1[strlen(s1)] = s2[i];
break;
}
}
}
}
void addprod(char *s)
{
int i;
prod[count][0] = s[0];
for(i=3;s[i]!='\0';i++)
{
if(!IsCap(s[i]))
add(ter,s[i]);
prod[count][i-2] = s[i];
}
prod[count][i-2] = '\0';
add(nt,s[0]);
count++;
}
void findfirst()
{
int i,j,n,k,e,n1;
for(i=0;i<count;i++)
{
for(j=0;j<count;j++)
{
n = findpos(prod[j][0]);
if(prod[j][1] == (char)238)
```

```
eps[n] = 1;
else
{
for(k=1,e=1;prod[j][k]!='\0' && e==1;k++)
{
if(!IsCap(prod[j][k]))
{
e=0;
add(first[n],prod[j][k]);
}
else
{
n1 = findpos(prod[j][k]);
addarr(first[n],first[n1]);
if(eps[n1] == 0)
e=0;
}
                        }
if(e==1)
eps[n]=1;
}
}
}
}
void findfollow()
{
int i,j,k,n,e,n1;
n = findpos(prod[0][0]);
add(follow[n],'$');
```

```
for(i=0;i<count;i++)
{
for(j=0;j<count;j++)
{
k = strlen(prod[j])-1;
for(;k>0;k--)
{
if(IsCap(prod[j][k]))
{
n=findpos(prod[j][k]);
if(prod[j][k+1] == '\0')     // A -> aB
{
n1 = findpos(prod[j][0]);
addarr(follow[n],follow[n1]);
}
if(IsCap(prod[j][k+1]))              // A -> aBb
{
n1 = findpos(prod[j][k+1]);
addarr(follow[n],first[n1]);
if(eps[n1]==1)
{
n1=findpos(prod[j][0]);
addarr(follow[n],follow[n1]);
}
}
else if(prod[j][k+1] != '\0')
add(follow[n],prod[j][k+1]);
}
}
```

```
}
}
}
void main()
{
char s[max],i;
printf("\nEnter the productions(type 'end' at the last of the production)\n");
scanf("%s",s);
while(strcmp("end",s))
{
addprod(s);
scanf("%s",s);
}
findfirst();
findfollow();
for(i=0;i<strlen(nt);i++)
{
printf("%c\t",nt[i]);
printf("%s",first[i]);
if(eps[i]==1)
printf("%c\t",(char)238);
else
printf("\t");
printf("%s\n",follow[i]);
}
getch();
}
```

- Consider the grammar

$S \rightarrow (L) \mid a$

$L \rightarrow L , S \mid S$

- Compute the function FIRST for all non terminals

FIRST(S) = { (, a)

FIRST(L) = FIRST(S) = { (, a)

$S \rightarrow (L) \mid a$

$L \rightarrow L , S \mid S$

- Compute the function FOLLOW for all non terminals

FOLLOW(S) = { **$** , **FOLLOW(L)** }

= { **$** , **)** , **,** }

FOLLOW(L) = { **)** , **,** }

- Consider the grammar

$S \rightarrow L = R \mid R$

$L \rightarrow * R \mid \mathbf{id}$

$R \rightarrow L$

- Compute the function FIRST for all non terminals

FIRST(S) = FIRST(L) and FIRST(R)

FIRST(L) = { * , **id**)

FIRST(R) = FIRST(L) = { * , **id** }

Therefore FIRST(S) = { * , **id** }

- Consider the grammar

$S \rightarrow L = R \mid R$

$L \rightarrow * R \mid \mathbf{id}$

$R \rightarrow L$

Compute the function FOLLOW for all non terminals

FOLLOW(S) = { **$** }

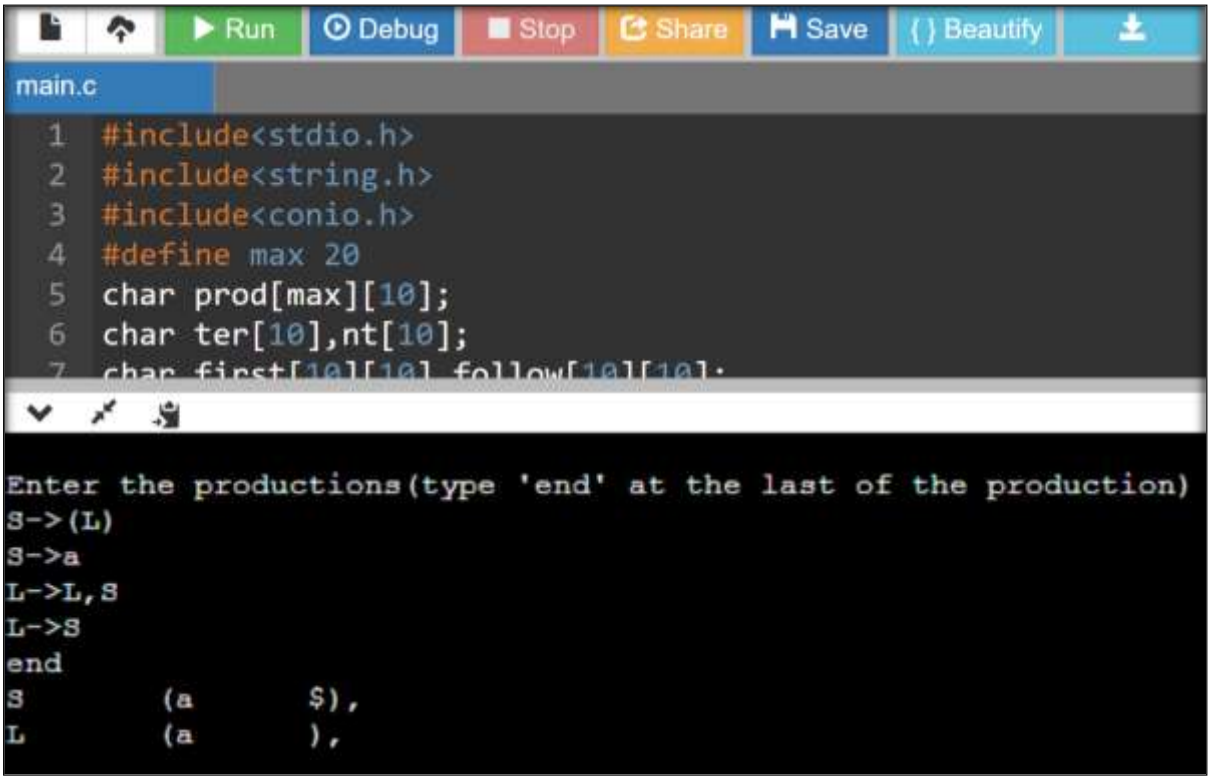
FOLLOW(L) = { **=** , **FOLLOW(R)** }

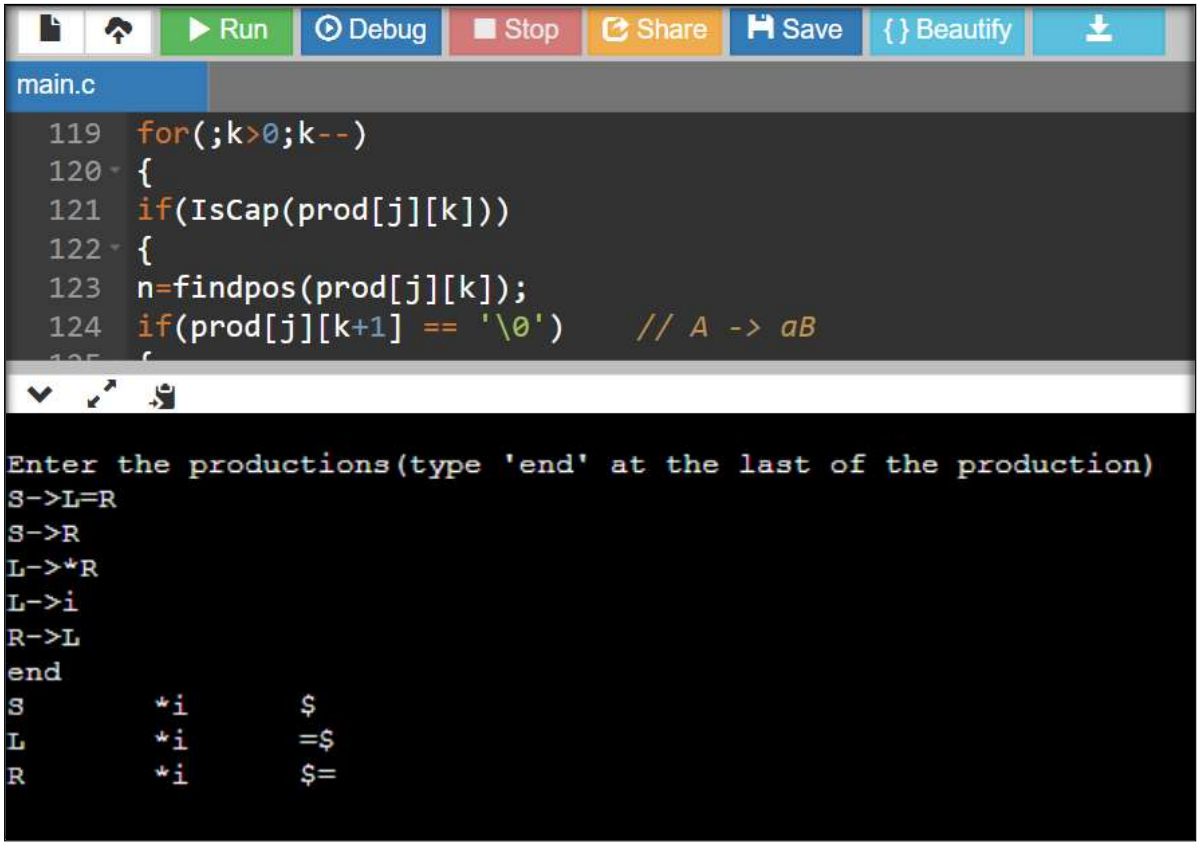
= { **=** , **$** }

FOLLOW(R) = { FOLLOW(S) & FOLLOW(L) }

= { **$** , **=** }

OUTPUT:


Run
Debug
Stop
Share
Save
{ } Beautify
main.c
1 #include<stdio.h>
2 #include<string.h>
3 #include<conio.h>
4 #define max 20
5 char prod[max][10];
6 char ter[10],nt[10];
Enter the productions(type 'end' at the last of the production)
S->(L)
S->a
L->L,S
L->S
end
S (a $),
L (a),


Run
Debug
Stop
Share
Save
{ } Beautify
main.c
119 for(;k>0;k--)
120 {
121 if(IsCap(prod[j][k]))
122 {
123 n=findpos(prod[j][k]);
124 if(prod[j][k+1] == '\0') // A -> aB
Enter the productions(type 'end' at the last of the production)
S->L=R
S->R
L->*R
L->i
R->L
end
S *i $
L *i =$
R *i $=

RESULT: The program for finding first and follow is successfully compiled and executed

CHITRALEKHA.CH

RA1911003010387