**COMPILER DESIGN PRACTICAL LAB-6**

**Predictive Parsing Table**

SANDRA MARIA TONY RA1911026010045 K1

**AIM:** A program for construction of **Predictive Parsing Table**

**THEORY:**

**Step 1:**First check for [left recursion](https://www.geeksforgeeks.org/removing-direct-and-indirect-left-recursion-in-a-grammar/) in the grammar, if there is left recursion in the grammar remove that and go to step 2.

**Step 2:**Calculate First() and Follow() for all non-terminals.

1. [**First**](https://www.geeksforgeeks.org/first-set-in-syntax-analysis/)**():** If there is a variable, and from that variable, if we try to drive all the strings then the beginning Terminal Symbol is called the First.
2. [Follow](https://www.geeksforgeeks.org/follow-set-in-syntax-analysis/)(): What is the Terminal Symbol which follows a variable in the process of derivation.

**Step 3:**For each production A –> α. (A tends to alpha)

1. Find First(α) and for each terminal in First(α), make entry A –> α in the table.
2. If First(α) contains ε (epsilon) as terminal than, find the Follow(A) and for each terminal in Follow(A), make entry A –> α in the table.
3. If the First(α) contains ε and Follow(A) contains $ as terminal, then make entry A –> α in the table for the $.

**ALGORITHM:**

1. Start

2. Ask the user to enter the set of productions. 3. Calculate First () and Follow () for all non-terminals.

4. Construct the parsing table.

5. Display the output.

6. Exit

**CODE:**

#include<stdio.h>

#include<conio.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 coordinates\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 pos\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: j++;

}

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

{

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 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]!='\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++;

}

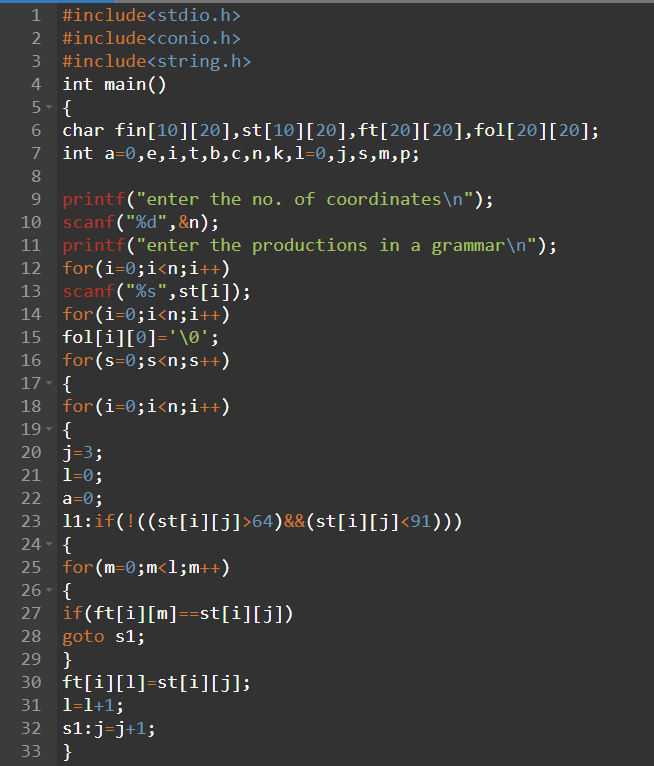
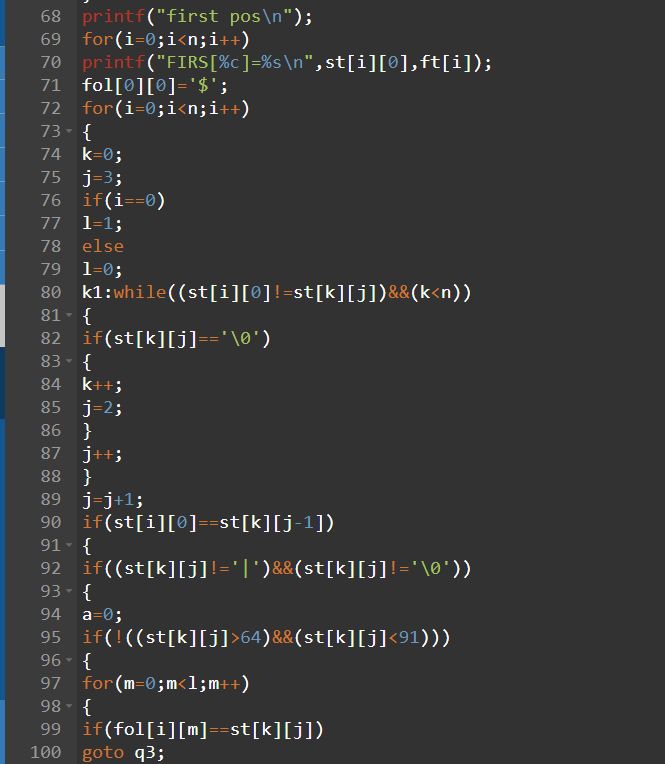
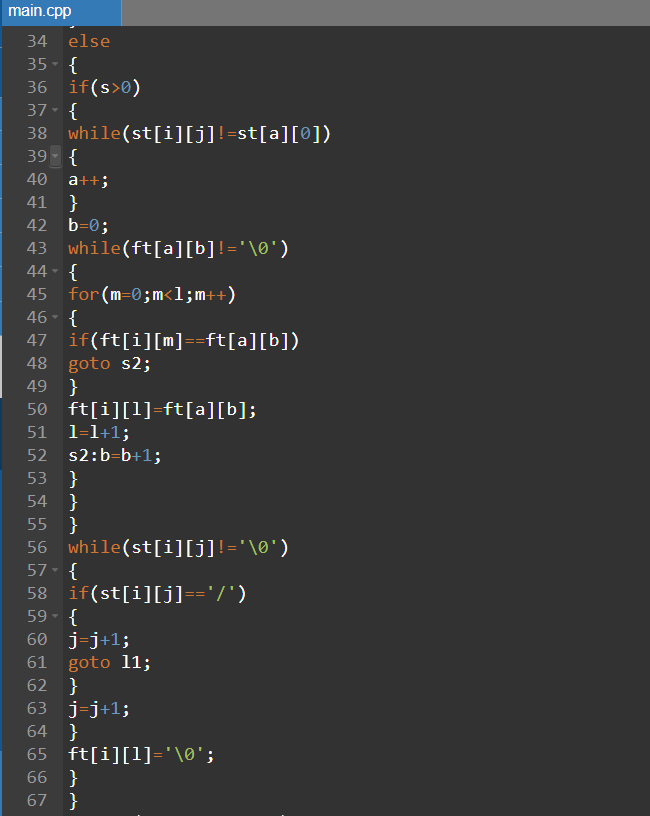
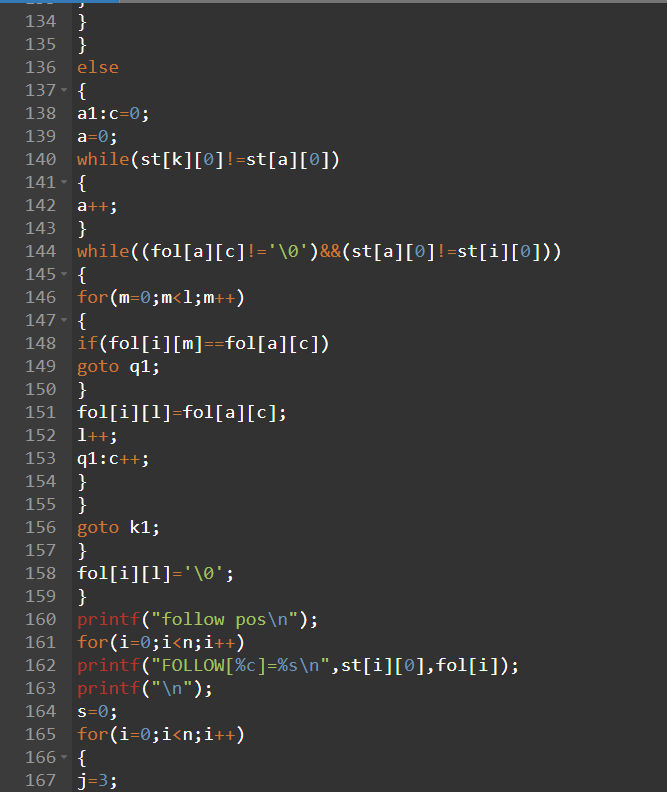
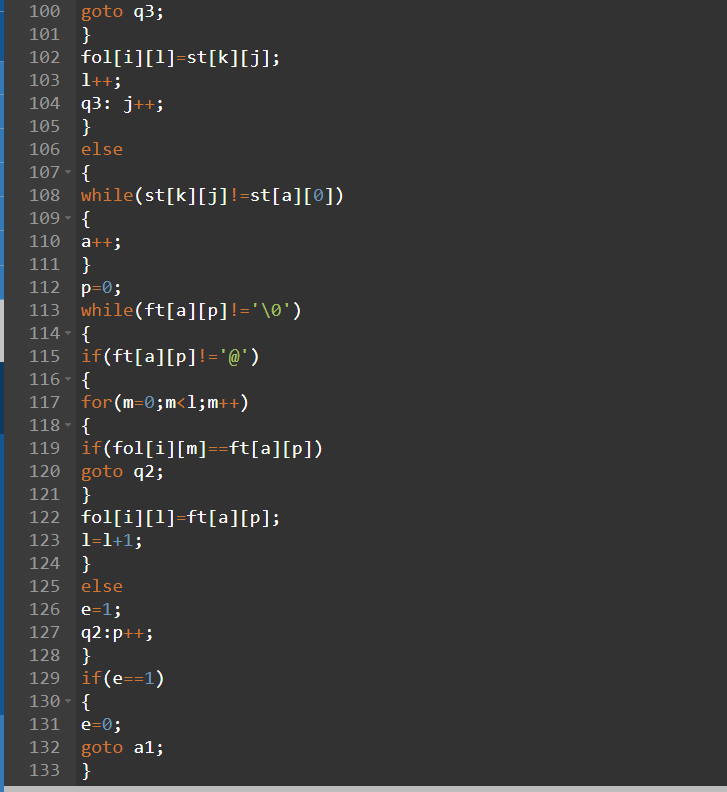
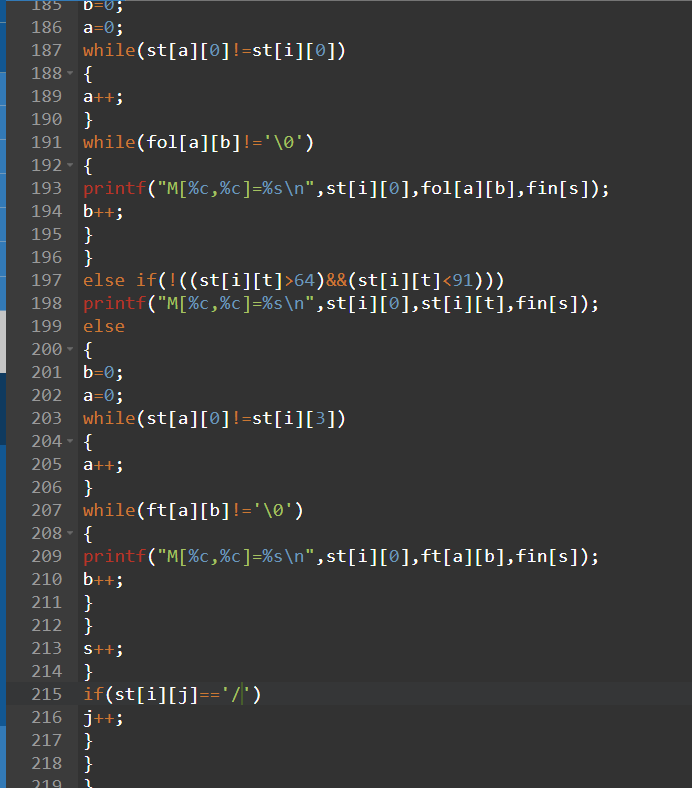
if(st[i][j]=='/')

j++;

}

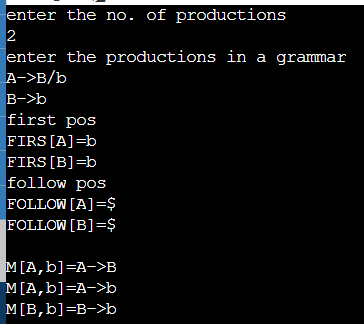
}

}

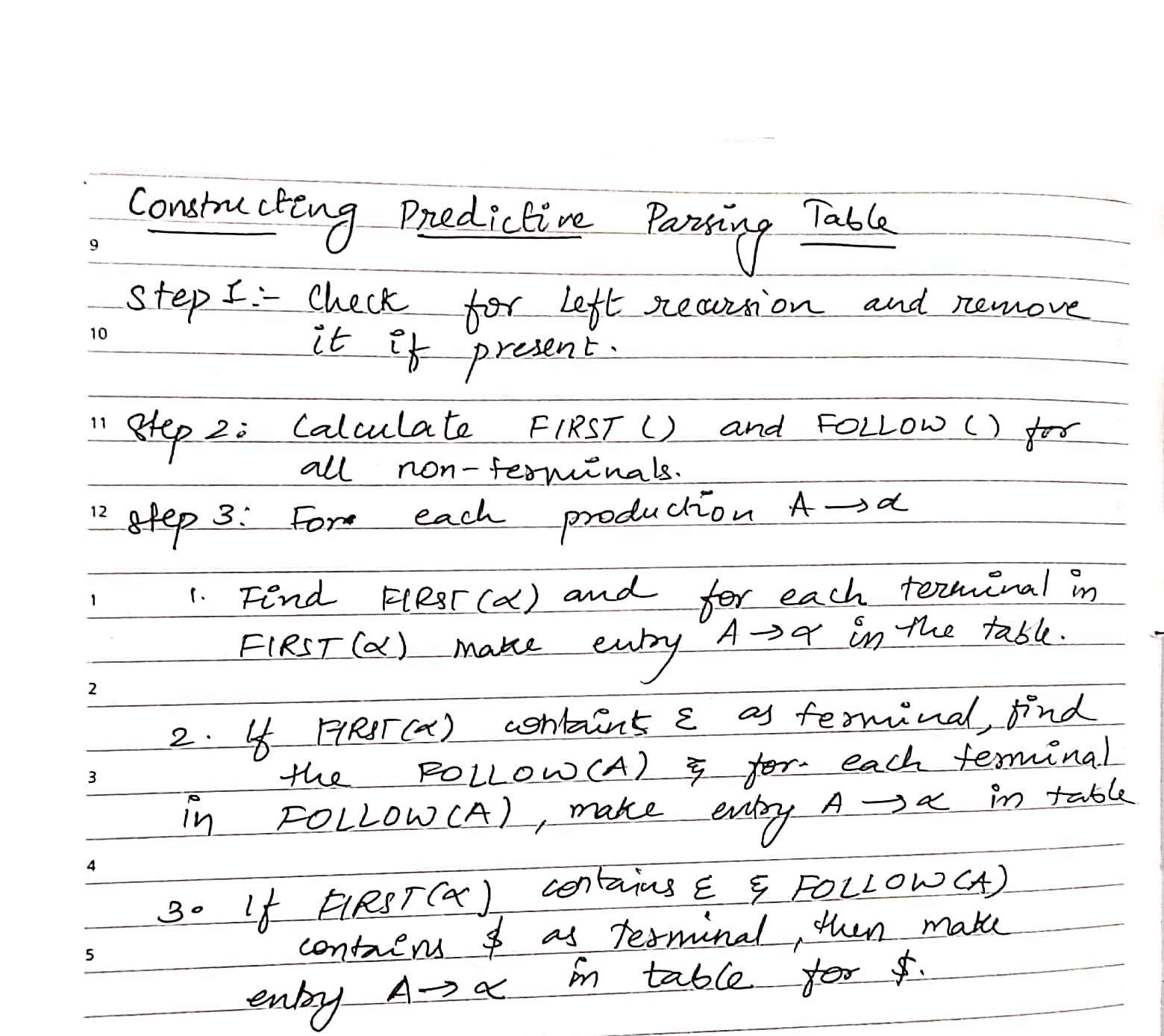
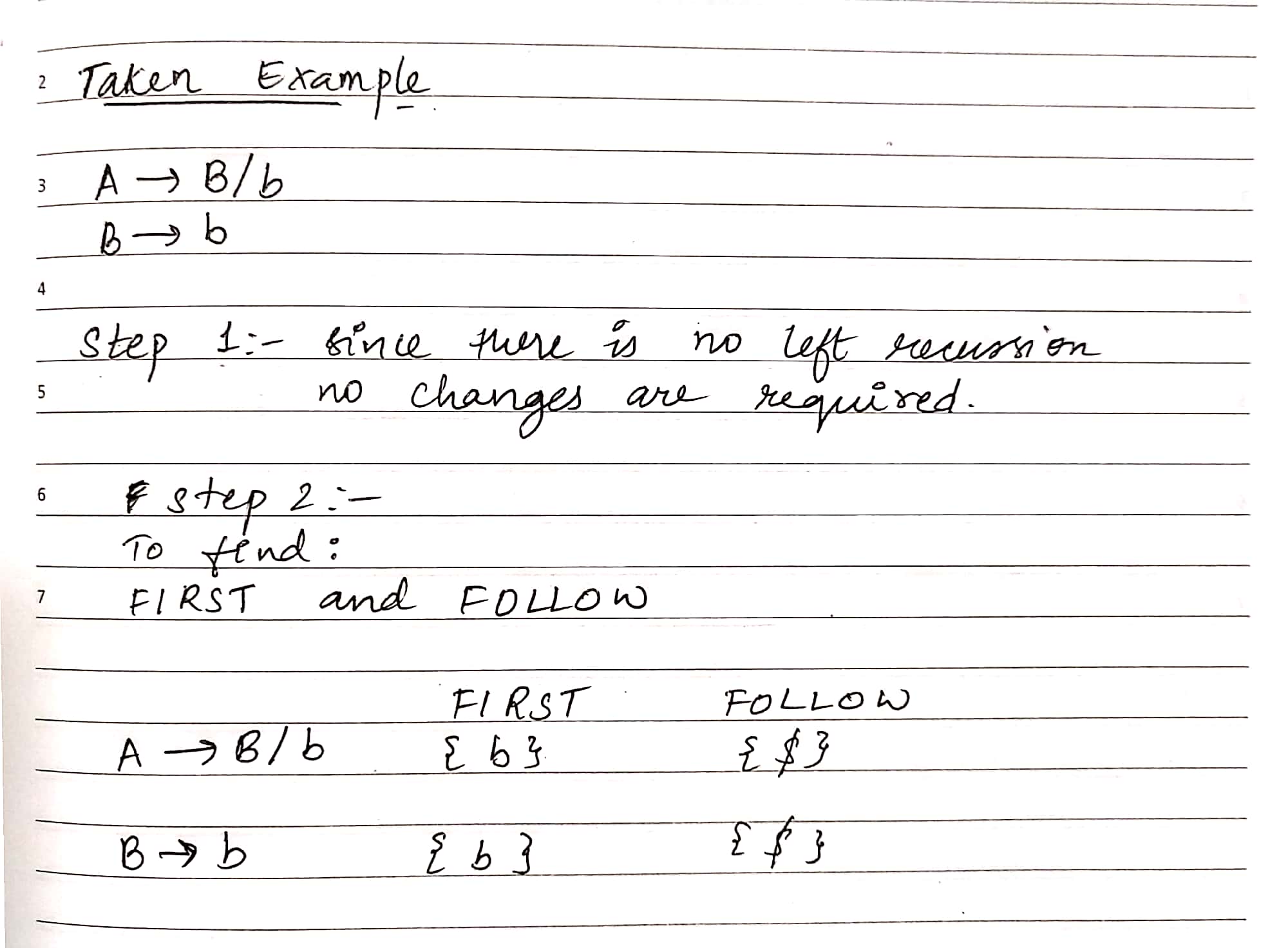
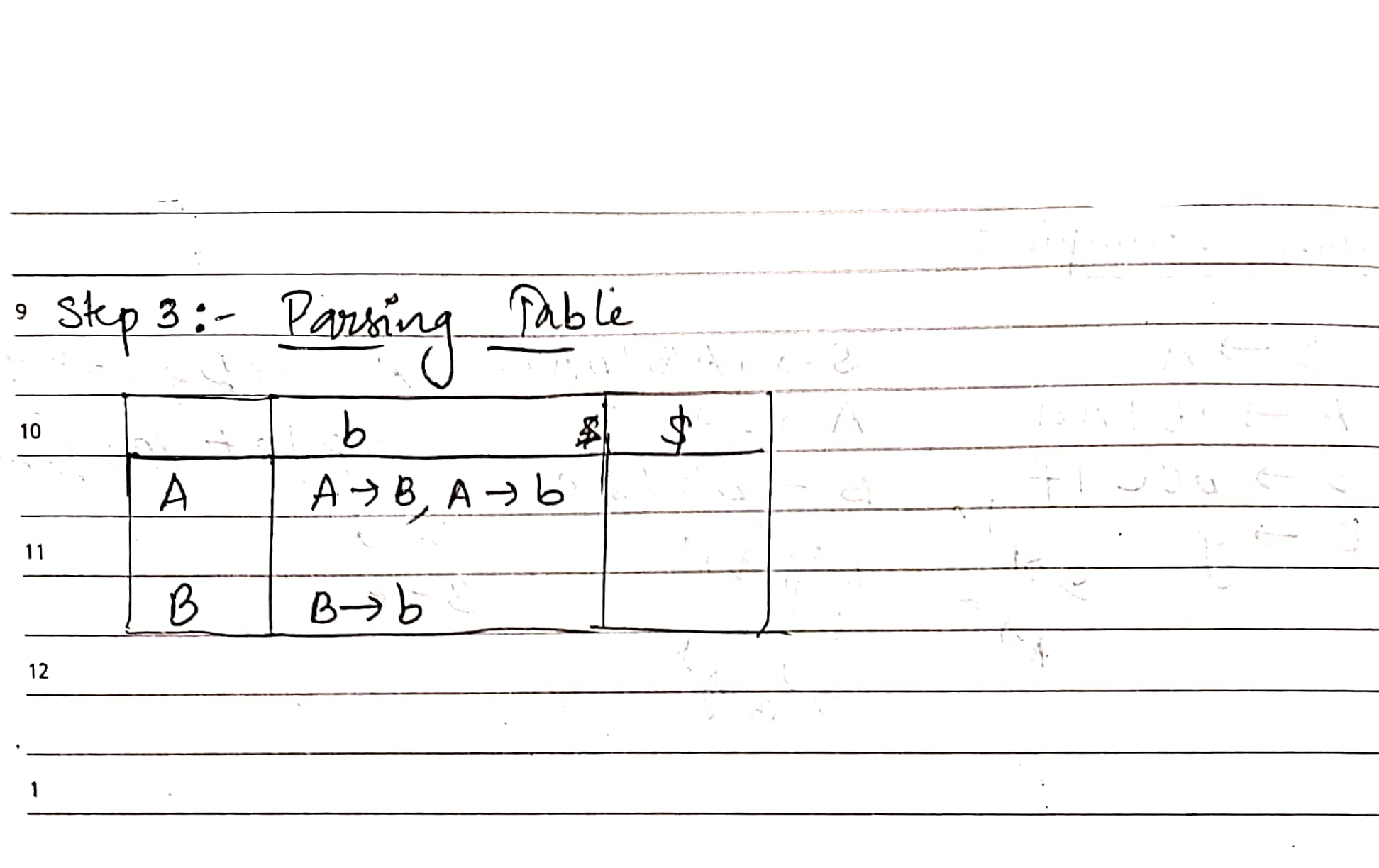
   

**Output:**

Here epsilon is taken as ‘@’.



**MANUAL CALCULATION:**

****  

kjhkjh

**RESULT:** The code for construction of **Predictive Parsing Table** produced same output as the result from manual calculation thus the program was executed successfully.