**Date:** 24/03/22

**Predictive Parsing Table**

*Ayush Jindal*

***RA1911003010308***

***Aim****: To write a program that outputs the predictive parsing table for given input grammar.*

***Algorithm:***

1. *Start the program.*
2. *Initialize the required variables.*
3. *Get the number of Non-terminals and the productions from the user as input.*
4. *For each input, 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];*

*}*

1. *Print the resulting stack.*
2. *Print if the grammar is accepted or not.*
3. *Exit the program*

***Code:***

*#include<stdio.h>*

*#include<conio.h>*

*#include<string.h>*

*void main()*

*{*

*char fin[10][20], st[10][20], first[20][20], follow[20][20];*

*int a=0,e,i,t,b,c,n,k,l=0,j,s,m,p;*

*printf("Enter the no. of Non-terminals\n");*

*scanf("%d",&n);*

*printf("Enter the productions in the grammar\n");*

*for(i=0;i<n;i++)*

*scanf("%s",st[i]);*

*for(i=0;i<n;i++)*

*follow[i][0]='\0'; // making all follow to null*

*for(s=0;s<n;s++) // Computation of FIRST*

*{*

*for(i=0;i<n;i++)*

*{*

*j=3;*

*l=0;*

*a=0;*

*l1:if(!((st[i][j]>64)&&(st[i][j]<91))) // if the element is not in 'A' to 'Z'*

*{*

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

*{*

*if(first[i][m]==st[i][j])*

*goto s1;*

*}*

*first[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(first[a][b]!='\0')*

*{*

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

*{*

*if(first[i][m]==first[a][b])*

*goto s2;*

*}*

*first[i][l]=first[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;*

*}*

*first[i][l]='\0';*

*}*

*}*

*printf("\n---------FIRST Computation----------\n");*

*for(i=0;i<n;i++)*

*{*

*printf("FIRST(%c)= {",st[i][0]);*

*j=0;*

*while(first[i][j+1]!='\0')*

*{*

*printf("%c,",first[i][j]);*

*j+=1;*

*}*

*printf("%c}\n",first[i][j]);*

*}*

*printf("\n");*

*follow[0][0]='$'; // Making follow of start state to $*

*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(follow[i][m]==st[k][j])*

*goto q3;*

*}*

*follow[i][l]=st[k][j];*

*l++;*

*q3:;*

*}*

*else*

*{*

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

*{*

*a++;*

*}*

*p=0;*

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

*{*

*if(first[a][p]!='#')*

*{*

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

*{*

*if(follow[i][m]==first[a][p])*

*goto q2;*

*}*

*follow[i][l]=first[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((follow[a][c]!='\0')&&(st[a][0]!=st[i][0]))*

*{*

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

*{*

*if(follow[i][m]==follow[a][c])*

*goto q1;*

*}*

*follow[i][l]=follow[a][c];*

*l++;*

*q1:c++;*

*}*

*}*

*goto k1;*

*}*

*follow[i][l]='\0';*

*}*

*printf("-------FOLLOW Computation------- \n");*

*for(i=0;i<n;i++)*

*{*

*printf("FOLLOW(%c)= {",st[i][0]);*

*j=0;*

*while(follow[i][j+1]!='\0')*

*{*

*printf("%c,",follow[i][j]);*

*j+=1;*

*}*

*printf("%c}\n",follow[i][j]);*

*}*

*printf("\n");*

*printf("------Predictive Parsing Table------\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(follow[a][b]!='\0')*

*{*

*printf("M[%c,%c]=%s\n",st[i][0],follow[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(first[a][b]!='\0')*

*{*

*printf("M[%c,%c]=%s\n",st[i][0],first[a][b],fin[s]);*

*b++;*

*}*

*}*

*s++;*

*}*

*if(st[i][j]=='|')*

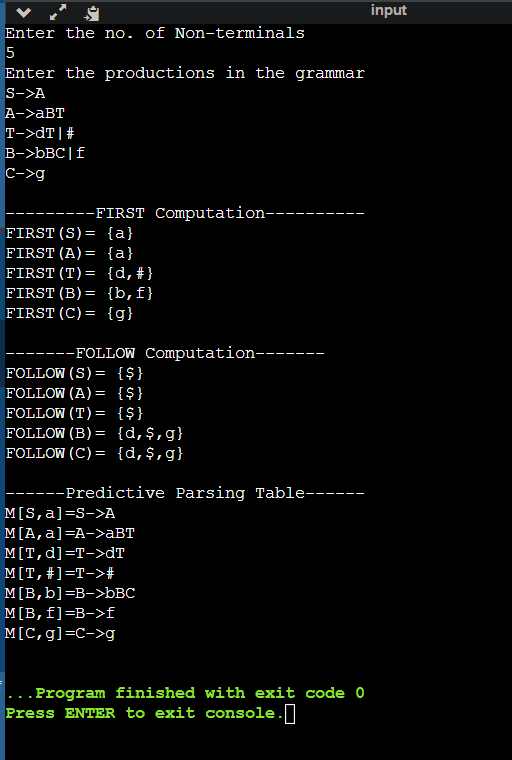
*j++;*

*}*

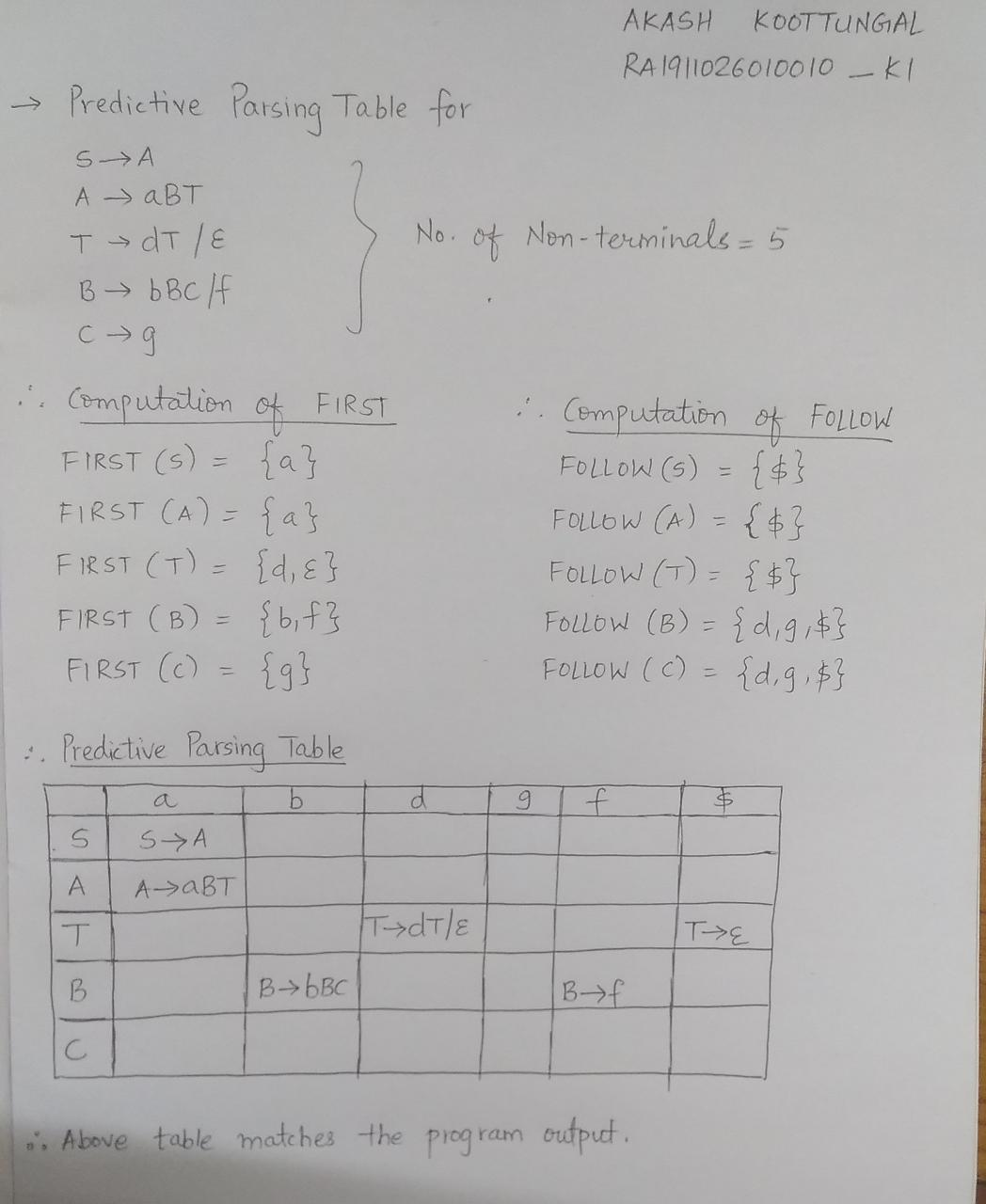
*}*

*}*

*Output:*

**

*Manual Solution:*

**

*Result: Thus, we learned to construct predictive parsing table for given grammar in C language.*