**COMPILER DESIGN PRACTICAL LAB-5**

**FIRST and FOLLOW Computation**

SANDRA MARIA TONY RA1911026010045 K1

**Computation of FIRST:**

**AIM:** A program for computation of FIRST.

**THEORY:**

**Rules to compute FIRST set:**

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)
   * If FIRST(Y1) contains Є then FIRST(X) = { FIRST(Y1) – Є } U { FIRST(Y2) }
   * If FIRST (Yi) contains Є for all i = 1 to n, then add Є to FIRST(X).

**ALGORITHM:**

1. Start

2. Ask the user to enter the set of productions

3. If x is a terminal, then FIRST(x) = { ‘x’ }

4 If x-> Є, is a production rule, then add Є to FIRST(x).

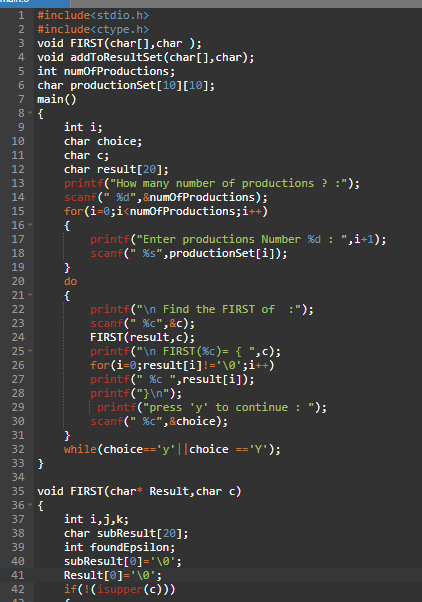
5. Display the output

6. Exit

**CODE:**

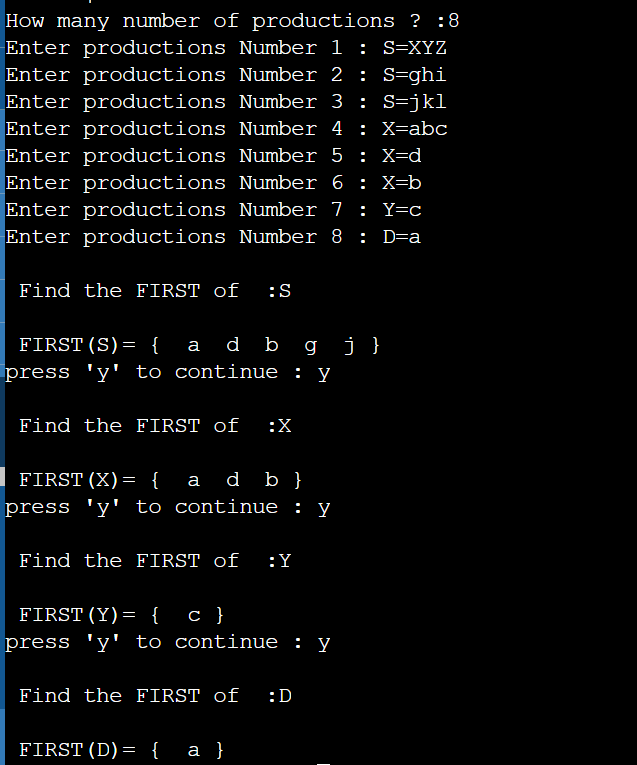
#include<stdio.h>  
#include<ctype.h>  
void FIRST(char[],char );  
void addToResultSet(char[],char);  
int numOfProductions;  
char productionSet[10][10];  
main()  
{  
    int i;  
    char choice;  
    char c;  
    char result[20];  
    printf("How many number of productions ? :");  
    scanf(" %d",&numOfProductions);  
    for(i=0;i<numOfProductions;i++)    {  
        printf("Enter productions Number %d : ",i+1);  
        scanf(" %s",productionSet[i]);  
    }  
    do  
    {  
        printf("\n Find the FIRST of  :");  
        scanf(" %c",&c);  
        FIRST(result,c);   
        printf("\n FIRST(%c)= { ",c);  
        for(i=0;result[i]!='\0';i++)  
        printf(" %c ",result[i        printf("}\n");  
         printf("press 'y' to continue : ");  
        scanf(" %c",&choice);  
    }  
    while(choice=='y'||choice =='Y');  
}  
void FIRST(char\* Result,char c)  
{  
    int i,j,k;  
    char subResult[20];  
    int foundEpsilon;  
    subResult[0]='\0';  
    Result[0]='\0';  
       if(!(isupper(c)))  
    {  
        addToResultSet(Result,c);  
               return ;  
    }    for(i=0;i<numOfProductions;i++)  
    {        if(productionSet[i][0]==c)  
        { if(productionSet[i][2]=='$') addToResultSet(Result,'$');  
                 else  
            {  
                j=2;  
                while(productionSet[i][j]!='\0')  
                {  
                foundEpsilon=0;  
                FIRST(subResult,productionSet[i][j]);  
                for(k=0;subResult[k]!='\0';k++)  
                    addToResultSet(Result,subResult[k]);  
                 for(k=0;subResult[k]!='\0';k++)  
                     if(subResult[k]=='$')  
                     {  
                         foundEpsilon=1;  
                         break;  
                     }                 if(!foundEpsilon)  
                     break;  
             j++; }}}  
     return ;  
}  
void addToResultSet(char Result[],char val)  
{  
    int k;  
    for(k=0 ;Result[k]!='\0';k++)  
        if(Result[k]==val)  
            return;  
    Result[k]=val;  
    Result[k+1]='\0';  
}

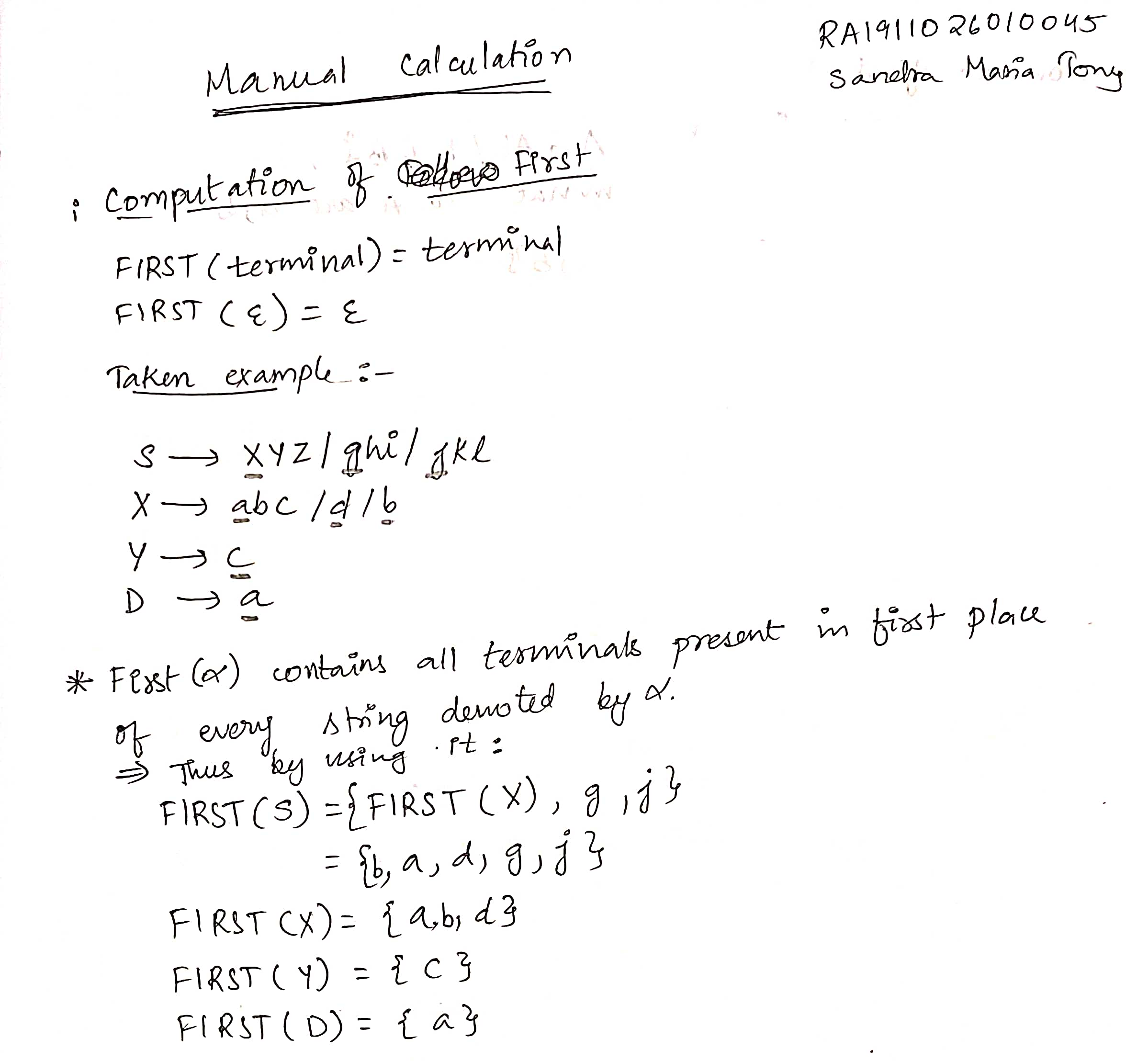
**CODE SCREENSHORT:**





**OUTPUT:**



**MANUAL CALCULATION:** ****

**RESULT:**

A program of computation of FIRST was compiled and executed successfully.

**Computation of FOLLOW**

**AIM:** A program for computation Of FOLLOW.

**THEORY:**

**Rules to compute FOLLOW set:**

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)

**ALGORITHM:**

1. Start

2. Ask the user to enter the set of productions

3. 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).

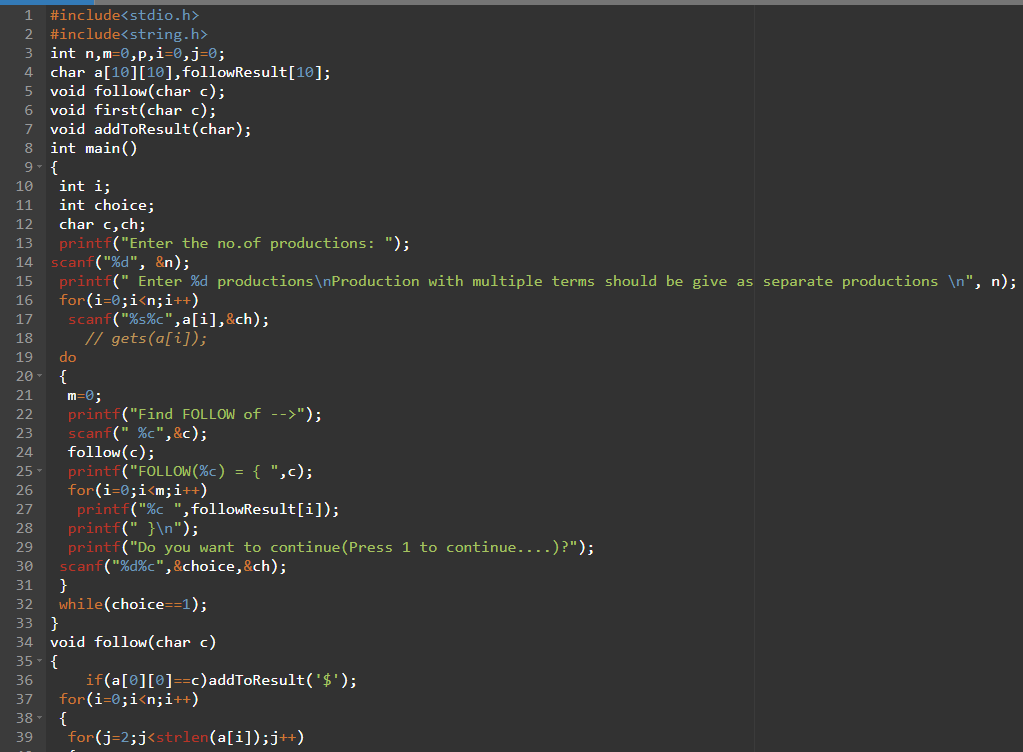
4. If A->pB is a production, then everything in FOLLOW(A) is in FOLLOW(B).

5. Display the output

6. Exit

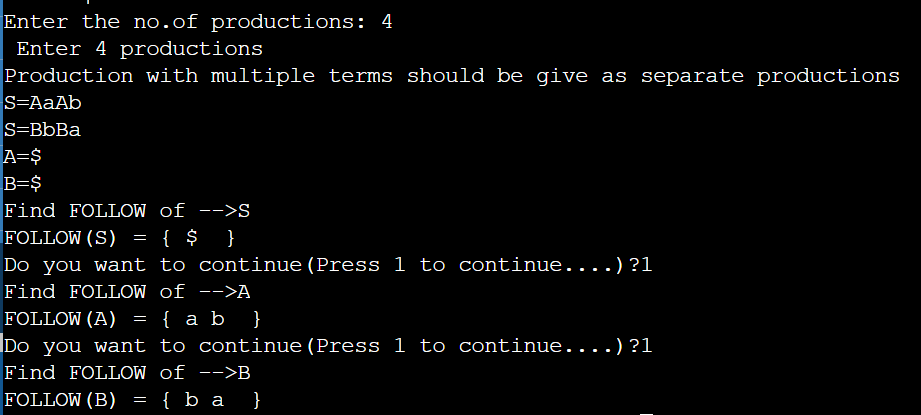
**CODE:**

#include<stdio.h>   
#include<string.h>  
int n,m=0,p,i=0,j=0;  
char a[10][10],followResult[10];  
void follow(char c);  
void first(char c);  
void addToResult(char);  
int main()  
{  
 int i;  
 int choice;  
 char c,ch;  
 printf("Enter the no.of productions: ");  
scanf("%d", &n);  
 printf(" Enter %d productions\nProduction with multiple terms should be give as separate productions \n", n);  
 for(i=0;i<n;i++)  
  scanf("%s%c",a[i],&ch);  
    // gets(a[i]);  
 do  
 {  
  m=0;  
  printf("Find FOLLOW of -->");  
  scanf(" %c",&c);  
  follow(c);  
  printf("FOLLOW(%c) = { ",c);  
  for(i=0;i<m;i++)  
   printf("%c ",followResult[i]);  
  printf(" }\n");  
  printf("Do you want to continue(Press 1 to continue....)?");  
 scanf("%d%c",&choice,&ch);  
 }  
 while(choice==1);  
}  
void follow(char c)  
{  
    if(a[0][0]==c)addToResult('$');  
 for(i=0;i<n;i++)  
 {  
  for(j=2;j<strlen(a[i]);j++)  
  {  
   if(a[i][j]==c)  
   {  
    if(a[i][j+1]!='\0')first(a[i][j+1]);  
    if(a[i][j+1]=='\0'&&c!=a[i][0])  
     follow(a[i][0]);  
   }  
  }  
 }  
}  
void first(char c)  
{  
      int k;  
                 if(!(isupper(c)))  
                     //f[m++]=c;  
                     addToResult(c);  
                 for(k=0;k<n;k++)  
                 {  
                 if(a[k][0]==c)  
                 {  
                 if(a[k][2]=='$') follow(a[i][0]);  
                 else if(islower(a[k][2]))  
                     //f[m++]=a[k][2];  
                     addToResult(a[k][2]);  
                 else first(a[k][2]);  
                 }  
                 }  
}  
void  addToResult(char c)  
{  
    int i;  
    for( i=0;i<=m;i++)  
        if(followResult[i]==c)  
            return;  
   followResult[m++]=c;  
}

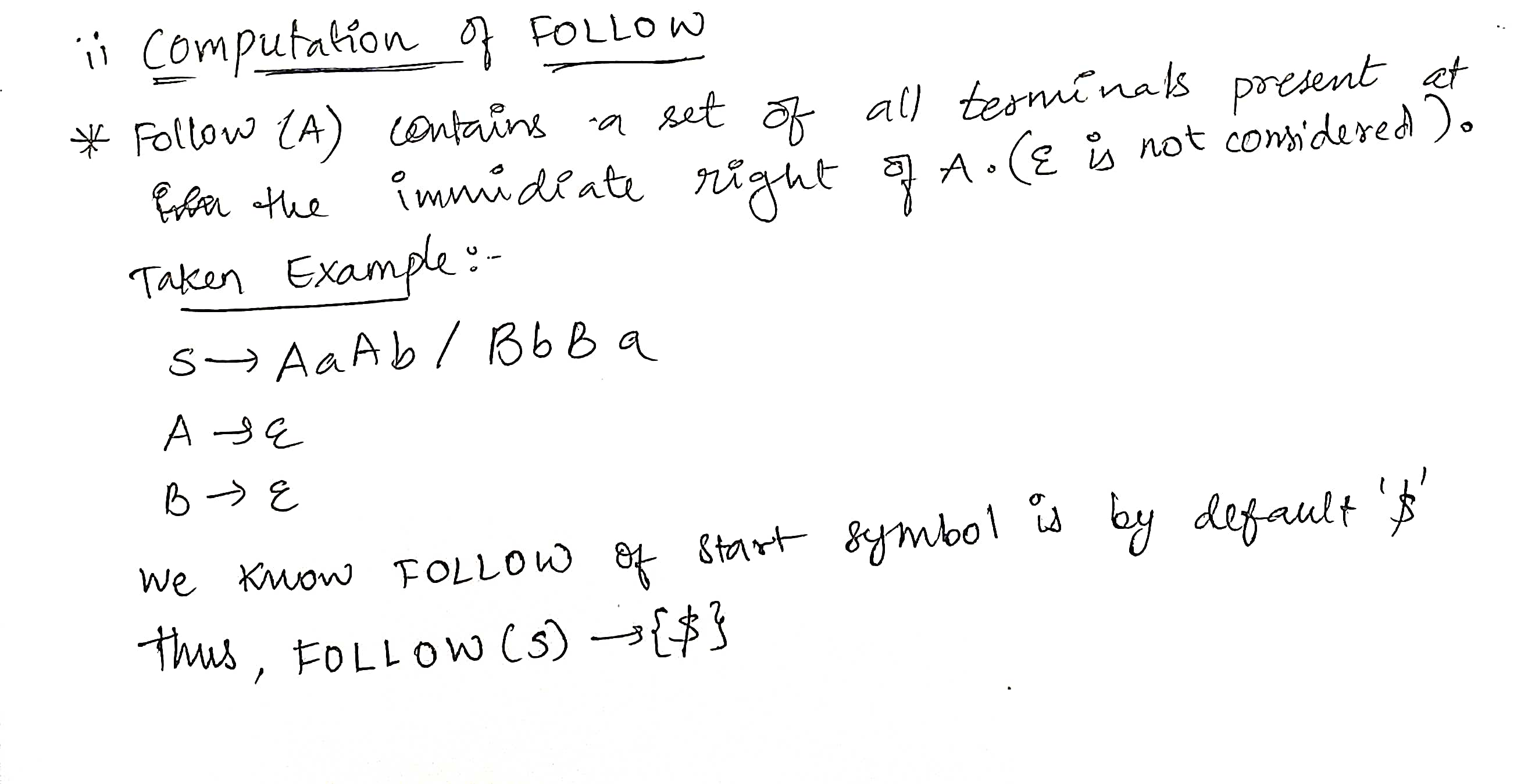
**CODE SCREENSHORT:** 

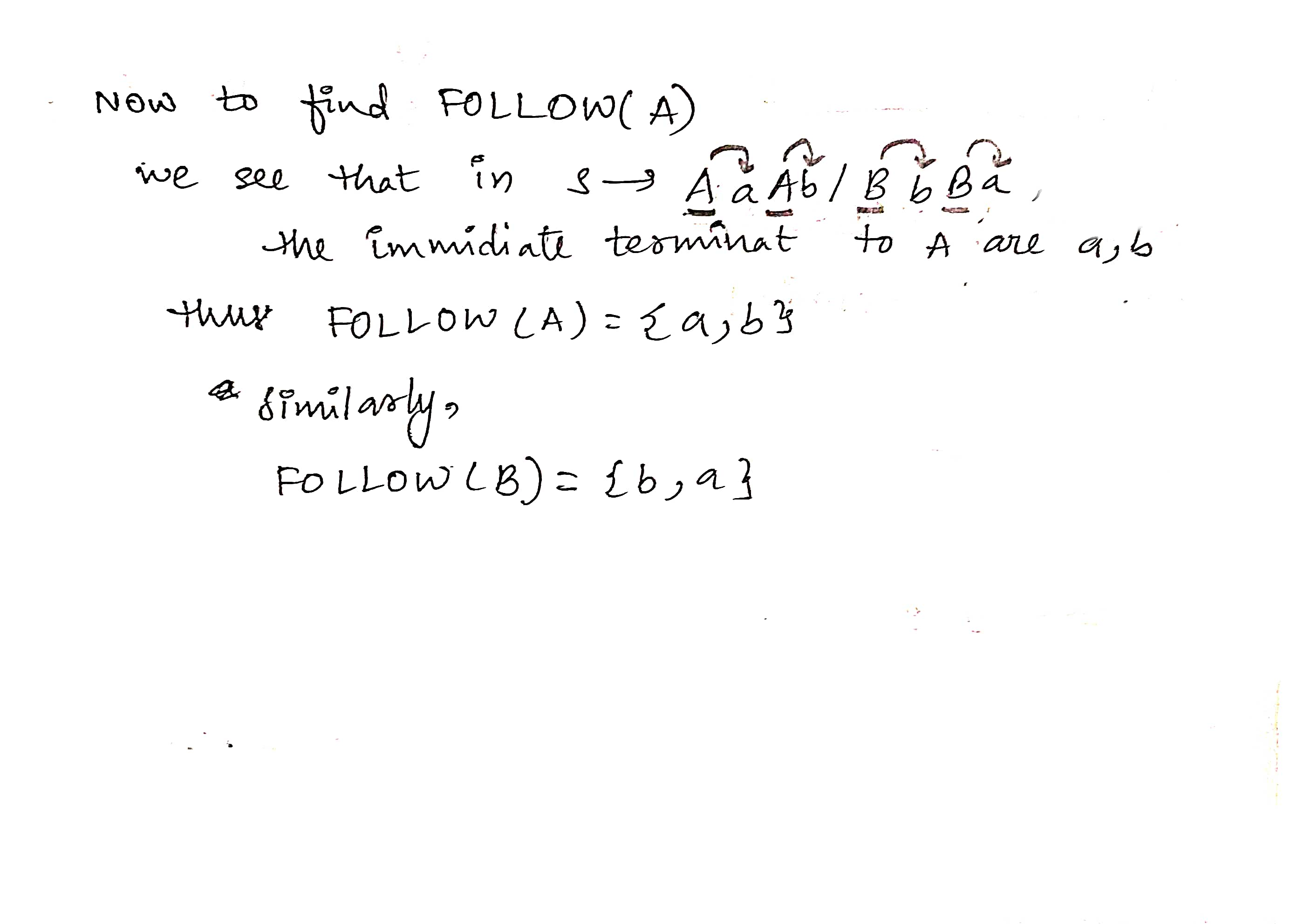


**OUTPUT:**



**MANUAL CALCULATION:**

****



**RESULT:**

A program of computation of FOLLOW was executed successfully.