Exp. No. 7

Write a C program to find FIRST( ) - predictive parser for the given grammar

S → AaAb / BbBa

A → ∈

B → ∈

Program:

```
#include<stdio.h>
#include<ctype.h>
void FIRST(char[],char );
void addToResultSet(char[],char);
int numOfProductions;
char productionSet[10][10];
int 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++)//read production string eg: E=E+T
 {
 printf("Enter productions Number %d : ",i+1);
 scanf(" %s",productionSet[i]);
 }
 do
 {
 printf("\n Find the FIRST of :");
 scanf(" %c",&c);
 FIRST(result,c); //Compute FIRST; Get Answer in 'result' array
 printf("\n FIRST(%c)= { ",c);
 for(i=0;result[i]!='\0';i++)
 printf(" %c ",result[i]); //Display result
 printf("}\n");
 printf("press 'y' to continue : ");
 scanf(" %c",&choice);
 }
 while(choice=='y'||choice =='Y');
}
/*
*Function FIRST:
```

```
*Compute the elements in FIRST(c) and write them
*in Result Array.
*/
void FIRST(char* Result,char c)
{
 int i,j,k;
 char subResult[20];
 int foundEpsilon;
 subResult[0]='\0';
 Result[0]='\0';
 //If X is terminal, FIRST(X) = {X}.
 if(!(isupper(c)))
 {
 addToResultSet(Result,c);
 return ;
 }
 //If X is non terminal
 //Read each production
 for(i=0;i<numOfProductions;i++)
 {
//Find production with X as LHS
 if(productionSet[i][0]==c)
 {
//If X → ε is a production, then add ε to FIRST(X).
if(productionSet[i][2]=='$') addToResultSet(Result,'$');
 //If X is a non-terminal, and X → Y1 Y2 … Yk
 //is a production, then add a to FIRST(X)
 //if for some i, a is in FIRST(Yi),
 //and ε is in all of FIRST(Y1), …, FIRST(Yi-1).
 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;
```

```
 }
 //No ε found, no need to check next element
 if(!foundEpsilon)
 break;
 j++;
 }
 }
 }
}
 return ;
}
/* addToResultSet adds the computed
*element to result set.
*This code avoids multiple inclusion of elements
 */
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';
}
```

OUTPUT:

```
How many number of productions ? :4
Enter productions Number 1 : S=AaAb
Enter productions Number 2 : S=BbBa
Enter productions Number 3 : A=$
Enter productions Number 4 : B=$

 Find the FIRST of :S

 FIRST(S)= {  $  a  b }
press 'y' to continue : y

 Find the FIRST of :A

 FIRST(A)= {  $ }
press 'y' to continue : y

 Find the FIRST of :B

 FIRST(B)= {  $ }
press 'y' to continue : n
```