Practical No. 9

Aim: Write a program to find FOLLOW set of a given grammar.

Requirement: GCC

Theory:

FOLLOW set is defined for each non-terminal X in the grammar. It contains the terminal symbols appearing just to the right of non-terminal X in the body of some production. Following rules are applied to compute FOLLOW (X):

- Place $ in FOLLOW (S), where S is the starting non-terminal and $ is the end-of-string character.
- If there is a production A → αBβ,
  then FOLLOW (B) = FIRST (β).
- If there is a production A → αB, or a production A → αBβ, where FIRST (β) contains ε
  then FOLLOW (B)=FOLLOW (A).

For eg.
E → T E’
E’ → + T E’ | ε
T → F T’
T’ → * F T’ | ε
F → (E) | id

FOLLOW (E) = {), $}
FOLLOW (E’) = {), $}
FOLLOW (T) = {+,), $}
FOLLOW (T’) = {+,), $}
FOLLOW (F) = {*, +,), $}

Program:

```
#include<stdio.h>
#include<string.h>
#include<ctype.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\nEnter each production in a form 'E=E+T'.\nUse '$'
		to represent epsilon.\nProduction with multiple terms should be give as separate
		productions \n", n);
		for(i=0;i<n;i++)
		{
			scanf("%s%c",a[i],&ch);
		}
		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;
        }
```

Output:

//Paste a color printout of the output here.

Conclusion:

Thus, a program to find the FOLLOW set of all the non-terminals in the given grammar is implemented successfully.