**Exp. No. 7**

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

S → AaAb / BbBa

A → ∈

B → ∈

**Program:**

#include<stdio.h>

#include<ctype.h>

#include<string.h>

int limit, x = 0;

char production[10][10], array[10];

void find\_first(char ch);

void find\_follow(char ch);

void Array\_Manipulation(char ch);

int main()

{

int count;

char option, ch;

printf("\nEnter Total Number of Productions:\t");

scanf("%d", &limit);

for(count = 0; count < limit; count++)

{

printf("\nValue of Production Number [%d]:\t", count + 1);

scanf("%s", production[count]);

}

do

{

x = 0;

printf("\nEnter production Value to Find Follow:\t");

scanf(" %c", &ch);

find\_follow(ch);

printf("\nFollow Value of %c:\t{ ", ch);

for(count = 0; count < x; count++)

{

printf("%c ", array[count]);

}

printf("}\n");

printf("To Continue, Press Y:\t");

scanf(" %c", &option);

}while(option == 'y' || option == 'Y');

return 0;

}

void find\_follow(char ch)

{

int i, j;

int length = strlen(production[i]);

if(production[0][0] == ch)

{

Array\_Manipulation('$');

}

for(i = 0; i < limit; i++)

{

for(j = 2; j < length; j++)

{

if(production[i][j] == ch)

{

if(production[i][j + 1] != '\0')

{

find\_first(production[i][j + 1]);

}

if(production[i][j + 1] == '\0' && ch != production[i][0])

{

find\_follow(production[i][0]);

}

}

}

}

}

void find\_first(char ch)

{

int i, k;

if(!(isupper(ch)))

{

Array\_Manipulation(ch);

}

for(k = 0; k < limit; k++)

{

if(production[k][0] == ch)

{

if(production[k][2] == '$')

{

find\_follow(production[i][0]);

}

else if(islower(production[k][2]))

{

Array\_Manipulation(production[k][2]);

}

else

{

find\_first(production[k][2]);

}

}

}

}

void Array\_Manipulation(char ch)

{

int count;

for(count = 0; count <= x; count++)

{

if(array[count] == ch)

{

return;

}

}

array[x++] = ch;

}

**Output:**

