Lab program 7:

7. Write a program for constructing LL(1) parsing.

Aim: Program to construct LL(1) parsing.

Algorithm:

1. If X=a=$, parser halts, string accepted.

2. If X=a !=$, parser pops X, and advances the input pointer to point to next input symbol.

3. If X is a nonterminal, the program consults entry M[X,a] of the parsing table M. Replace the top of stack(X) with production rule cossesponding to entry in table.If entry = ERROR, call error recovery routine.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
char s[20],stack[20];
void main()
{
    char m[5][6][3]={"tb"," "," ","tb"," "," "," ","+tb"," "," ","n","n","fc"," "," ","fc"," "," "," "
","n","*fc","
                                a","n","n","i"," "," ","(e)"," "," "};
    int size[5][6]={2,0,0,2,0,0,0,3,0,0,1,1,2,0,0,2,0,0,0,1,3,0,1,1,1,0,0,3,0,0};
    int i,j,k,n,str1,str2;
    clrscr();
    printf("\n Enter the input string: ");
    scanf("%s",s);
    strcat(s,"$");
    n=strlen(s);
    stack[0]='$';
    stack[1]='e';
    i=1;
    j=0;
    printf("\nStack Input\n");
    printf("_____\n");
    while((stack[i]!='$')&&(s[j]!='$'))
    {
        if(stack[i]==s[j])
        {
            i--;
            j++;
        }
        switch(stack[i])
        {
            case 'e': str1=0;
            break;
            case 'b': str1=1;
            break;
```

```
            case 't': str1=2;
            break;
            case 'c': str1=3;
           break;
           case 'f': str1=4;
           break;
        }
        switch(s[j])
        {
            case 'i': str2=0;
            break;
            case '+': str2=1;
            break;
            case '*': str2=2;
            break;
            case '(': str2=3;
            break;
            case ')': str2=4;
            break;
            case '$': str2=5;
            break;
        }
        if(m[str1][str2][0]=='\0')
        {
            printf("\nERROR");
            exit(0);
        }
        else if(m[str1][str2][0]=='n')
             i--;
        else
            if(m[str1][str2][0]=='i')
                 stack[i]='i';
            else
           {
                 for(k=size[str1][str2]-1;k>=0;k--)
                 {
                      stack[i]=m[str1][str2][k];
                      i++;
                 }
                  i--;
            }
        for(k=0;k<=i;k++)
             printf(" %c",stack[k]);
        printf(" ");
        for(k=j;k<=n;k++)
             printf("%c",s[k]);
        printf(" \n ");
    }
    printf("\n SUCCESS");
    getch();
}
```

Output:

Enter the input string: i*i+i

Stack	INPUT
$bt	i*i+i$
$bcf	i*i+i$
$bci	i*i+i$
$bc	*i+i$
$bcf*	*i+i$
$bcf	i+i$
$bci	i+i$
$bc	+i$
$b	+i$
$bt+	+i$
$bt	i$
$bcf	i$
$ bci	i$
$bc	$
$b	$
$	$

success