Name: Aashutosh Kumar Pandit
Roll No: CH.EN.U4CSE22076

Lab-4

Title: To implement LL(1) parsing using C program.

Code:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char s[20], stack[20];

// Parsing table
char m[5][6][4] = {
    {"tb", " ",  " ",  "tb", " ",  " "},
    {" ",  "+tb"," ",  " ",  "n",  "n"},
    {"fc", " ",  " ",  "fc", " ",  " "},
    {" ",  "n",  "*fc"," ",  "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 main()
{
    int i, j, k, n, str1, str2;

    printf("\nEnter the input string: ");
    scanf("%s", s);
    strcat(s, "$");
    n = strlen(s);

    stack[0] = '$';
    stack[1] = 'e';
    i = 1;  // stack top pointer
    j = 0;  // input pointer

    printf("\nStack\tInput\n");
    printf("_____________\n\n");

    while ((stack[i] != '$') || (s[j] != '$'))
    {
        if (stack[i] == s[j])
        {
            i--;
```

```
            j++;
        }
        else
        {
            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;
                default:
                    printf("\nERROR: Invalid symbol on stack: %c\n", stack[i]);
                    exit(0);
            }

            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;
                default:
                    printf("\nERROR: Invalid input symbol: %c\n", s[j]);
                    exit(0);
            }

            if (m[str1][str2][0] == ' ')
            {
                printf("\nERROR: No rule found for %c on stack and %c in input\n", stack[i], s[j]);
                exit(0);
            }
            else if (m[str1][str2][0] == 'n')
            {
                i--;
            }
            else if (m[str1][str2][0] == 'i')
            {
                stack[i] = 'i';
            }
            else
            {
                int sz = size[str1][str2];
                i--;
                for (k = sz - 1; k >= 0; k--)
                {
                    stack[++i] = m[str1][str2][k];
                }
            }
        }

        for (k = 0; k <= i; k++)
            printf("%c", stack[k]);
        printf("\t");

        for (k = j; k < n; k++)
            printf("%c", s[k]);
        printf("\n");
    }

    if (stack[i] == '$' && s[j] == '$')
        printf("\nSUCCESS\n");
    else
        printf("\nERROR: Parsing failed.\n");

    return 0;
}
```

Output:

```
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/lab$ nano parser.c
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/lab$ gcc parser.c -o parser
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/lab$ ./parser

Enter the input string: i+i*i

Stack   Input
----------------

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

SUCCESS
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/lab$ ./parser

Enter the input string: i

Stack   Input
----------------

$bt     i$
$bcf    i$
$bci    i$
$bc     $
$b      $
$       $

SUCCESS
```