CHRIST
(DEEMED TO BE UNIVERSITY)
DELHI-NCR, INDIA

Data Structure and Algorithm (MCA 271)

Lab Practical –

BY

Himanshu Heda (24225013)

SUBMITTED TO

Prof. Vandna Kansal

SCHOOL OF SCIENCES

2024-2025

Program Description:

Code of the program

Output: - Paste the o/p of the program.

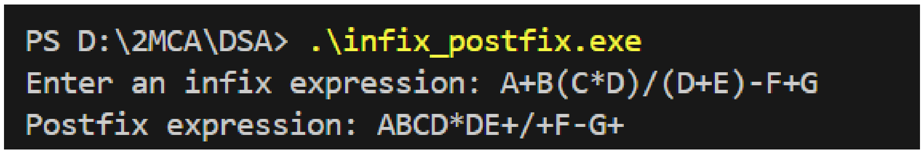
Write a C program for infix expression to postfix conversion using Stack.

```c
#include <stdio.h>

#define MAX 100

// Stack structure
typedef struct {
    int top;
    char items[MAX];
} Stack;

// Function to initialize the stack
void initStack(Stack* s) {
    s->top = -1;
}

// Function to check if the stack is empty
int isEmpty(Stack* s) {
    return s->top == -1;
}

// Function to push an item onto the stack
void push(Stack* s, char item) {
    if (s->top < MAX - 1) {
        s->items[++(s->top)] = item;
    } else {
        printf("Stack Overflow\n");
    }
}

// Function to pop an item from the stack
char pop(Stack* s) {
    if (!isEmpty(s)) {
        return s->items[(s->top)--];
    } else {
        printf("Stack Underflow\n");
        return '\0'; // Return a null character if stack is empty
    }
}
```

```
// Function to peek the top item of the stack
char peek(Stack* s) {
    if (!isEmpty(s)) {
        return s->items[s->top];
    }
    return '\0';
}

// Function to check the precedence of operators
int precedence(char op) {
    switch (op) {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        case '^':
            return 3;
        default:
            return 0;
    }
}

// Function to check if the character is an operator
int isOperator(char c) {
    return c == '+' || c == '-' || c == '*' || c == '/' || c == '^';
}

// Function to convert infix to postfix
void infixToPostfix(char* infix, char* postfix) {
    Stack s;
    initStack(&s);
    int j = 0;

    for (int i = 0; infix[i] != '\0'; i++) {
        char current = infix[i];

        if (current >= 'A' && current <= 'Z') { // If the character is an
operand (A-Z)
```

```
            postfix[j++] = current;
        } else if (current == '(') { // If the character is '('
            push(&s, current);
        } else if (current == ')') { // If the character is ')'
            while (!isEmpty(&s) && peek(&s) != '(') {
                postfix[j++] = pop(&s);
            }
            pop(&s); // Remove '(' from the stack
        } else if (isOperator(current)) { // If the character is an operator
            while (!isEmpty(&s) && precedence(peek(&s)) >=
precedence(current)) {
                postfix[j++] = pop(&s);
            }
            push(&s, current);
        }
    }

    // Pop all the operators from the stack
    while (!isEmpty(&s)) {
        postfix[j++] = pop(&s);
    }

    postfix[j] = '\0'; // Null-terminate the postfix expression
}

int main() {
    char infix[MAX], postfix[MAX];

    printf("Enter an infix expression: ");
    fgets(infix, MAX, stdin);

    // Remove newline character if present
    for (int i = 0; infix[i] != '\0'; i++) {
        if (infix[i] == '\n') {
            infix[i] = '\0';
            break;
        }
    }

    infixToPostfix(infix, postfix);
    printf("Postfix expression: %s\n", postfix);
    return 0;
}
```

OUTPUT : --



```
PS D:\2MCA\DSA> .\infix_postfix.exe
Enter an infix expression: A+B(C*D)/(D+E)-F+G
Postfix expression: ABCD*DE+/+F-G+
```