Institute of Computer Technology

B. Tech. Computer Science and Engineering

Sub: DS Branch: BDA Class: A

Name: Jatin Patel

Enrolment No.: 23162121029

Sem: 3

Class: A

Subject: DS

Practical: 04

Date: 21/08/24

# **Practical 4**

Stack infix to postfix Parishram is a 7th semester, who is studying at GUNI-ICT. During his “Compiler Design” course, his course faculty explained him that compiler work differently while it does evaluation of an expression due to below reasons:

Infix expressions are readable and solvable by humans because of easily distinguishable order of operators, but compiler doesn&#39;t have integrated order of operators.

To avoid this traversing, Infix expressions are converted to postfix expression before evaluation.

Make a program to convert infix expression into postfix using stack.

Sample Input:

(a-b)\*c+(d+f)

Sample Output:

ab-c\*df++

**Code:**

#include <stdio.h>

void push(char stack[], int \*top, char ch) {

    stack[++(\*top)] = ch;

}

char pop(char stack[], int \*top) {

    return stack[(\*top)--];

}

int main() {

    char infix[50], postfix[50], stack[50];

    int top = -1, i = 0, j = 0;

    printf("Enter an infix expression: ");

    scanf("%s", infix);

    while (infix[i] != '\0') {

        if (infix[i] == '(') {

            push(stack, &top, infix[i]);

        } else if (infix[i] == ')') {

            while (stack[top] != '(') {

                postfix[j++] = pop(stack, &top);

            }

            top--;

        } else if (infix[i] == '+' || infix[i] == '-') {

            while (top != -1 && (stack[top] == '+' || stack[top] == '-' || stack[top] == '\*' || stack[top] == '/')) {

                postfix[j++] = pop(stack, &top);

            }

            push(stack, &top, infix[i]);

        } else if (infix[i] == '\*' || infix[i] == '/') {

            while (top != -1 && (stack[top] == '\*' || stack[top] == '/')) {

                postfix[j++] = pop(stack, &top);

            }

            push(stack, &top, infix[i]);

        } else {

            postfix[j++] = infix[i];

        }

        i++;

    }

    while (top != -1) {

        postfix[j++] = pop(stack, &top);

    }

    postfix[j] = '\0';

    printf("Postfix expression: %s\n", postfix);

    return 0;

}

**Output:**

