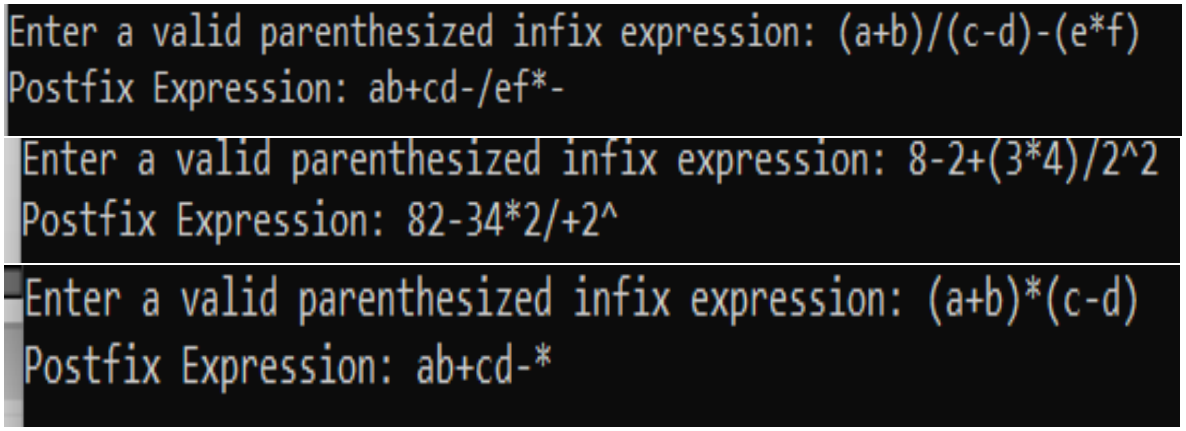
RAMITH H

1WN24CS222

PROGRAM 2 – CONVERSION OF INFIX TO POSTFIX EXPRESSION

```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#define MAX 100

char stack[MAX];
int top = -1;

void push(char c) {
    if (top == MAX - 1) {
        printf("Stack Overflow\n");
    } else {
        top = top + 1;
        stack[top] = c;
    }
}

char pop() {
    char val;
    if (top == -1) {
        printf("Stack Underflow\n");
        return -1;
    } else {
        val = stack[top];
        top = top - 1;
        return val;
    }
}

char peek() {
    if (top == -1)
        return '\0';
    return stack[top];
}

int precedence(char c) {
    if (c == '+' || c == '-') return 1;
    if (c == '*' || c == '/') return 2;
    return 0;
}
```

```c
void infixToPostfix(char infix[], char postfix[]) {
    int i, k = 0;
    char c;

    for (i = 0; infix[i] != '\0'; i++) {
        c = infix[i];
        if (isalnum(c)) {
            postfix[k] = c;
            k = k + 1;
        } else if (c == '(') {
            push(c);
        } else if (c == ')') {
            while (top != -1 && peek() != '(') {
                postfix[k] = pop();
                k = k + 1;
            }
            pop();
        } else {
            while (top != -1 && precedence(peek()) >= precedence(c)) {
                postfix[k] = pop();
                k = k + 1;
            }
            push(c);
        }
    }

    while (top != -1) {
        postfix[k] = pop();
        k = k + 1;
    }
    postfix[k] = '\0';
}

int main() {
    char infix[MAX], postfix[MAX];
    printf("Enter a valid parenthesized infix expression: ");
    scanf("%s", infix);
    infixToPostfix(infix, postfix);
    printf("Postfix Expression: %s\n", postfix);
    return 0;
}
```


Enter a valid parenthesized infix expression: (a+b)/(c-d)-(e*f)
Postfix Expression: ab+cd-/ef*-
Enter a valid parenthesized infix expression: 8-2+(3*4)/2^2
Postfix Expression: 82-34*2/+2^
Enter a valid parenthesized infix expression: (a+b)*(c-d)
Postfix Expression: ab+cd-*