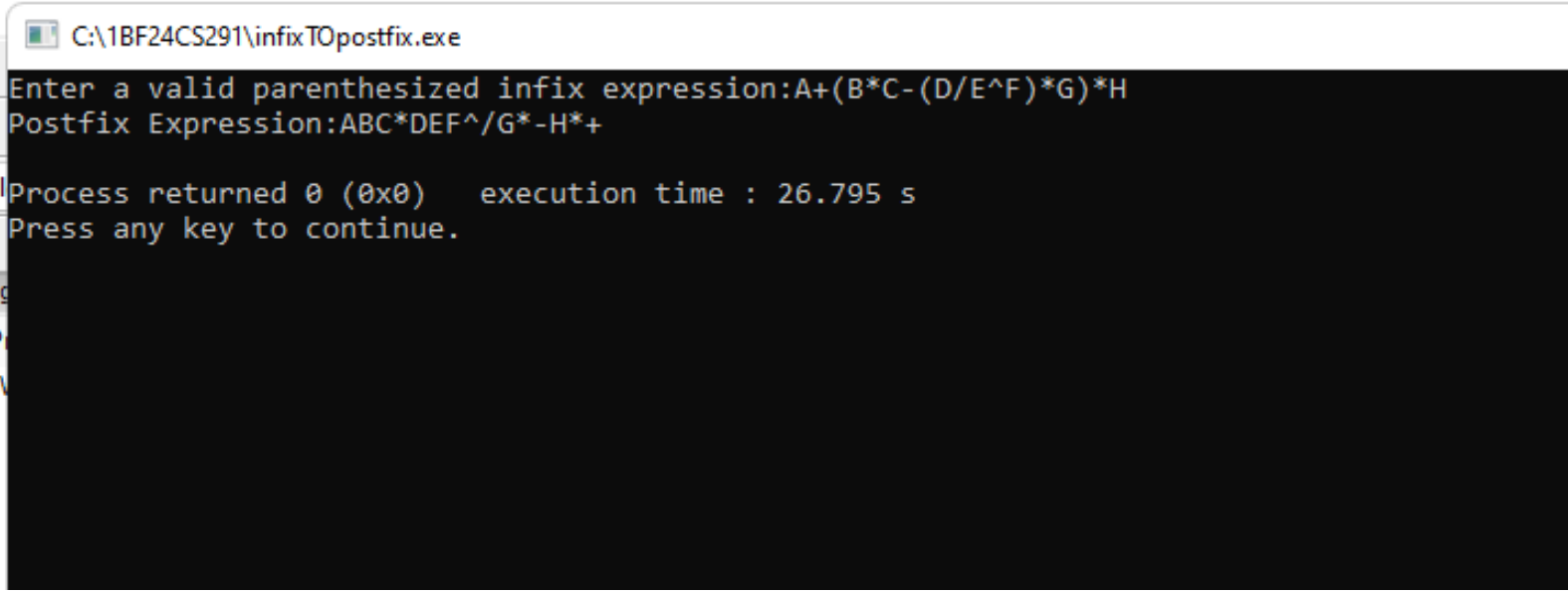
C:\1BF24CS291\infixTOpostfix.exe
Enter a valid parenthesized infix expression:A+(B*C-(D/E^F)*G)*H
Postfix Expression:ABC*DEF^/G*-H*+
Process returned 0 (0x0)   execution time : 26.795 s
Press any key to continue.

```
int associativity (char op) {
    if (op == '^') return 1;
    return 0;
}

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;
        }
        else if (c == '(') {
            push(c);
        }
        else if (c == ')') {
            while (peek() != '(') {
                postfix[k++] = pop();
            }
            pop();
        }
        else {
            while (top != -1 && (precedence(peek()) > precedence(c))
                || (precedence(peek()) == precedence(c) &&
                associativity(c) == 0)) {
                postfix[k++] = pop();
            }
            push(c);
        }
    }
    while (top != -1) {
        postfix[k++] = pop();
    }
    top--;
    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;
}
```

o/p:- Enter a valid parenthesized infix expression:
A+(B*C-(D/E^F)*G)*H

Postfix Expression : ABC*DEF^/G*-H*+

(a) Wap to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators

Pseudocode

Algorithm:-

1. start
2. Initialize an Empty stack of fixed size
3. Initilize an Empty array POSTFIX
4. Read the infix expression from Left to Right
5. Iterate through infix expression each character at a time.

Case 1:- if ch is digit or character append it to array POSTFIX

Case 2:- if ch is opening bracket, push it to stack.

Case 3:- if ch is closing bracket, pop it from stack and append it to postfix until corresponding opening bracket

Case 4:- if ch is an operator check at top of the Stack (i) if top of stack is operator of lower precedence than incoming operator append it to ~~POSTFIX array~~ stack

(ii) if top of stack is operator having higher precedence than incoming operator pop and append the top of stack to POSTFIX Expression & push incoming operator to stack

(iii) if incoming operator has higher precedence than top of stack push it into stack

(iv) if incoming operator has Equal precedence check associativity

* Associativity

L-R → pop & append operator of stack to postFix array & push incoming operator to stack

R-L → push incoming operator to stack

6. Print elements of POSTFIX Array
7. stop

MG 6/10/23

Code:-

```
#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");
        return;
    }
    stack[++top] = c;
}

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

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

int precedence(char op) {
    switch(op) {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        case '^':
            return 3;
        case '(':
            return 0;
    }
    return -1;
}
```