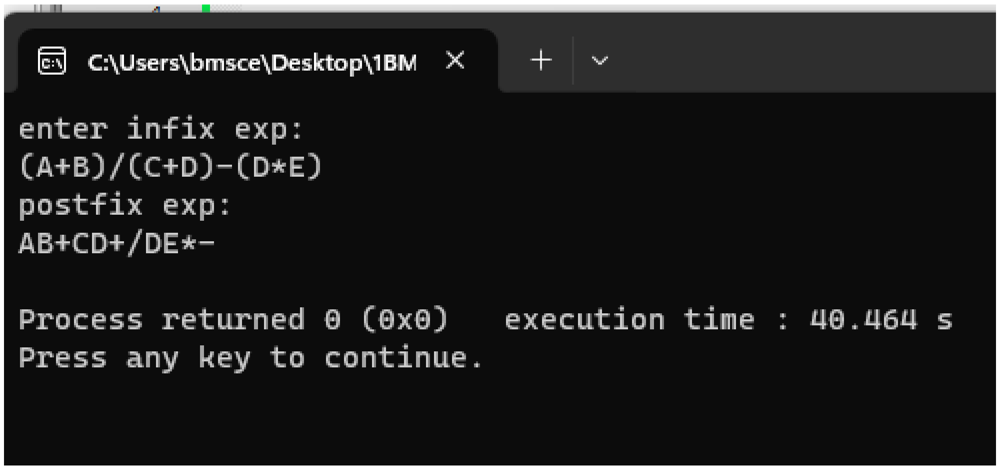
INFIX TO POSTFIX:

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

#define MAX 100

char stack[MAX];
char infix[MAX], postfix[MAX];
int top = -1;

void push(char);
char pop();
int isempty();
void infixToPostfix();
int space(char);
int print();
int precedence(char);

int main()
{
    printf("enter infix exp: \n");
    gets(infix);

    infixToPostfix();
    print();
    return 0;
}

void infixToPostfix()
{
    int i, j = 0;
    char next, symbol;
    for (i = 0; i < strlen(infix); i++)
    {
        symbol = infix[i];
        if (!space(symbol))
        {
            switch (symbol)
            {
            case '(':
                push(symbol);
                break;
            case ')':
                while ((next = pop()) != '(')
                {
                    postfix[j++] = next;
                }
                break;

            case '+':
            case '-':
            case '*':
            case '/':
            case '^':
                while (!isempty() && precedence(stack[top]) >= precedence(symbol))
                {
                    postfix[j++] = pop();
                }
                push(symbol);
                break;
            default:
                postfix[j++] = symbol;
            }
        }
    }

    while (!isempty())
    {
        postfix[j++] = pop();
    }
    postfix[j] = '\0';
}

int space(char c)
{
    if (c == ' ' || c == '\t')
    {
        return 1;
    }
    else
        return 0;
}

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

int print()
{
    int i = 0;
    printf("postfix exp: \n");
    while (postfix[i])
    {
        printf("%c", postfix[i++]);
    }
    printf("\n");
}

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

char pop()
{
    char c;
    if (top == -1)
    {
        printf("stack underflow\n");
        exit(1);
    }
    else
    {
        c = stack[top];
        top--;
        return c;
    }
}

isempty()
{
    if (top == -1)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
```

output:



```
enter infix exp:
(A+B)/(C+D)-(D*E)
postfix exp:
AB+CD+/DE*-

Process returned 0 (0x0)   execution time : 40.464 s
Press any key to continue.
```