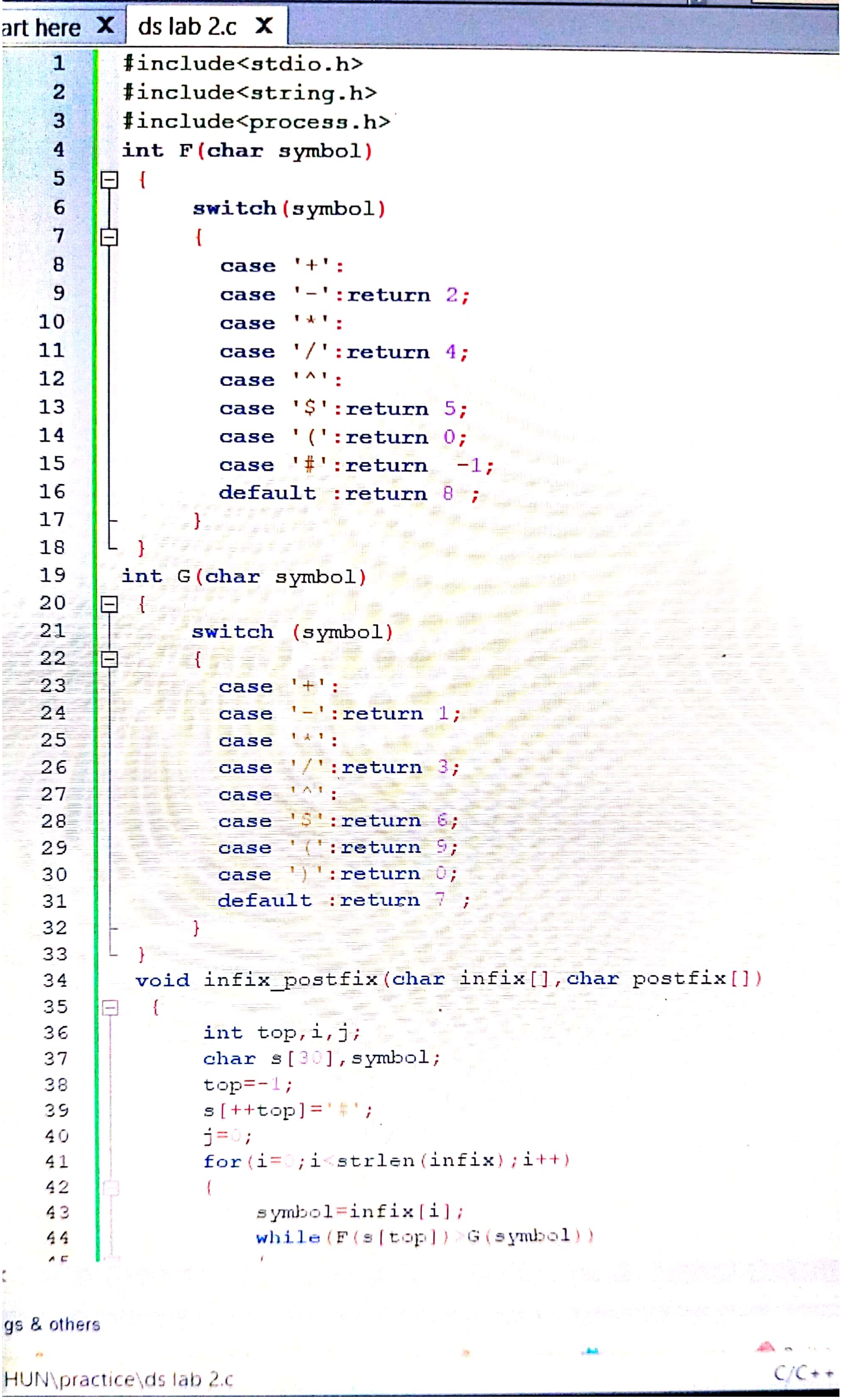
LAB 2

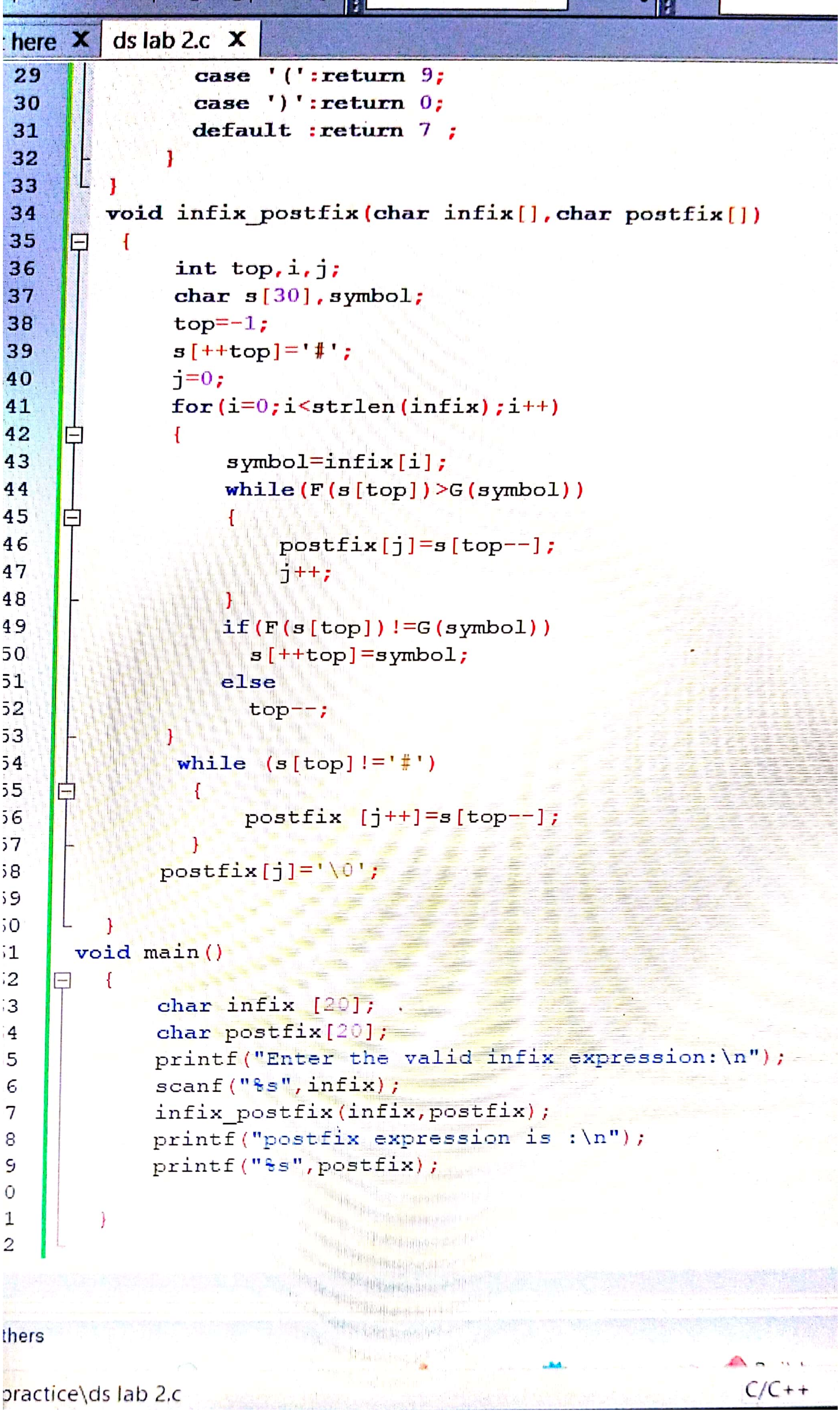
→ WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands & the binary operators +, -, * & /.

```
#include <stdio.h>
# include < string.h>
# include < process.h>
int F (char Symbol)
{  switch (symbol).
    {
       case '+' :
       case '-' : return 2;
       case '*' :
       case '/' : return 4;
       case '^' :
       case '$' : return 5;
       case '(' : return 0;
       case '#' : return -1;
       default : return 8;
    }
}
int G (char Symbol).
{
   switch (symbol)
     {
        case '+' :
        case '-' : return 1;
        case '*' :
        case '/' : return 3;
        case '^' :
        case '$' : return 6;
        case '(' : return 9;
        case ')' : return 0;
        default : return 7;
     }
}
```

```
void infix-postfix (char infix[], char postfix[])
{ int top, i, j;
  char s[30], symbol;
  top = -1;
  s[++top] = '#';
  j = 0;
  for (i = 0; i < strlen(infix); i++)
   {  symbol = infix[i];
      while (F(s[top]) > G(symbol))
      {    postfix[j] = s[top--];
           j++;
      }
      if (F(s[top]) ! = G(symbol))
       s[++top] = symbol;
      else
       top--;
   }
   while (s[top] ! = '#')
   {   postfix[j++] = s[top--];
   }
     postfix[j] = '\0';
}
   void main()
   { char infix [20];
     char postfix [20];
     printf("enter the valid infix expn \n");
      scanf ("%s", infix);
      infix-postfix (infix, postfix);
     printf("the postfix expn is \n");
     printf ("%s\n", postfix);
     ~~gtc~~
     getch ();
   }
```



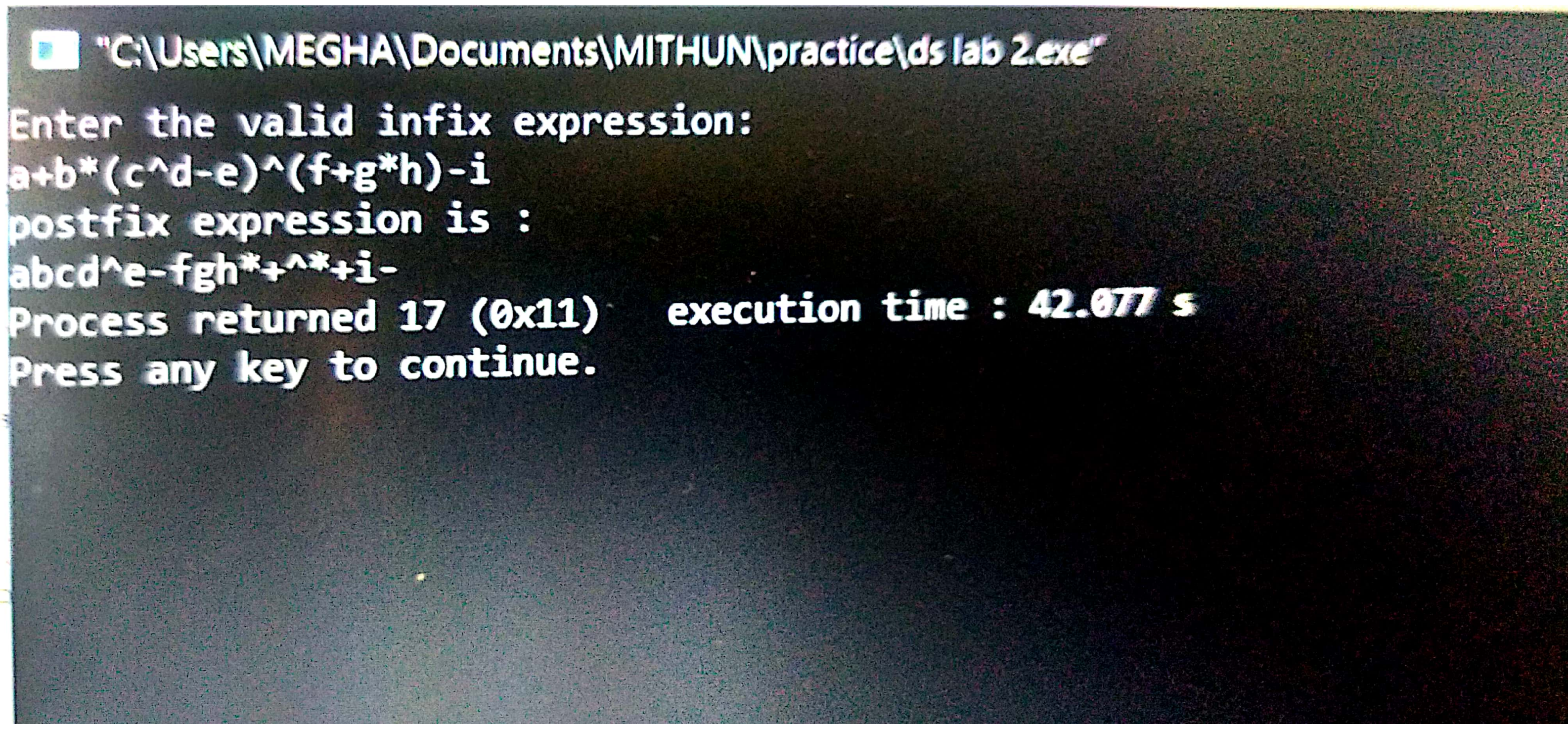
```c
#include<stdio.h>
#include<string.h>
#include<process.h>
int F(char symbol)
 {
        switch(symbol)
        {
          case '+':
          case '-':return 2;
          case '*':
          case '/':return 4;
          case '^':
          case '$':return 5;
          case '(':return 0;
          case '#':return  -1;
          default :return 8 ;
        }
 }
int G(char symbol)
 {
        switch (symbol)
        {
          case '+':
          case '-':return 1;
          case '*':
          case '/':return 3;
          case '^':
          case '$':return 6;
          case '(':return 9;
          case ')':return 0;
          default :return 7 ;
        }
 }
 void infix_postfix(char infix[],char postfix[])
  {
        int top,i,j;
        char s[30],symbol;
        top=-1;
        s[++top]='#';
        j=0;
        for(i=0;i<strlen(infix);i++)
        {
            symbol=infix[i];
            while(F(s[top])>G(symbol))
```



```c
            case '(':return 9;
            case ')':return 0;
            default :return 7 ;
        }
    }
    void infix_postfix(char infix[],char postfix[])
    {
        int top,i,j;
        char s[30],symbol;
        top=-1;
        s[++top]='#';
        j=0;
        for(i=0;i<strlen(infix);i++)
        {
            symbol=infix[i];
            while(F(s[top])>G(symbol))
            {
                postfix[j]=s[top--];
                j++;
            }
            if(F(s[top])!=G(symbol))
                s[++top]=symbol;
            else
                top--;
        }
        while (s[top]!='#')
        {
            postfix [j++]=s[top--];
        }
        postfix[j]='\0';

    }
void main()
    {
        char infix [20];
        char postfix[20];
        printf("Enter the valid infix expression:\n");
        scanf("%s",infix);
        infix_postfix(infix,postfix);
        printf("postfix expression is :\n");
        printf("%s",postfix);

    }
```



```
"C:\Users\MEGHA\Documents\MITHUN\practice\ds lab 2.exe"
Enter the valid infix expression:
a+b*(c^d-e)^(f+g*h)-i
postfix expression is :
abcd^e-fgh*+^*+i-
Process returned 17 (0x11)   execution time : 42.077 s
Press any key to continue.
```