#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

#define SIZE 100

char s[SIZE];

int top = -1;

void push(char ch)

{

if(top >= SIZE-1)

{

printf("\nStack Overflow.");

}

else

{

top++;

s[top] = ch;

}

}

char pop()

{

char ch ;

if(top <0)

{

printf("stack under flow: invalid infix expression");

getchar();

exit(1);

}

else

{

ch = s[top];

top--;

return(ch);

}

}

int operator(char op)

{

if(op == '^' || op == '\*' || op == '/' || op == '+' || op =='-')

{

return 1;

}

else

{

return 0;

}

}

int precedence(char ch)

{

if(ch == '^')

{

return(3);

}

else if(ch == '\*' || ch == '/')

{

return(2);

}

else if(ch == '+' || ch == '-')

{

return(1);

}

else

{

return(0);

}

}

void InfixToPostfix(char infix[], char postfix[])

{

int i=0, j=0;

char ch;

char x;

push('(');

strcat(infix,")");

ch=infix[i];

while(ch != '\0')

{

if(ch == '(')

{

push(ch);

}

else if( isdigit(ch) || isalpha(ch))

{

postfix[j] = ch;

j++;

}

else if(operator(ch) == 1)

{

x=pop();

while(operator(x) == 1 && precedence(x)>= precedence(ch))

{

postfix[j] = x;

j++;

x = pop();

}

push(x);

push(ch);

}

else if(ch == ')')

{

x = pop();

while(x != '(')

{

postfix[j] = x;

j++;

x = pop();

}

}

else

{

printf("\nInvalid infix Expression.\n");

getchar();

exit(1);

}

i++;

ch = infix[i];

}

if(top>0)

{

printf("\nInvalid infix Expression.\n");

getchar();

exit(1);

}

if(top>0)

{

printf("\nInvalid infix Expression.\n");

getchar();

exit(1);

}

postfix[j] = '\0';

}

void main()

{

char infix[SIZE], postfix[SIZE];

printf("\nEnter Infix expression : ");

gets(infix);

InfixToPostfix(infix,postfix);

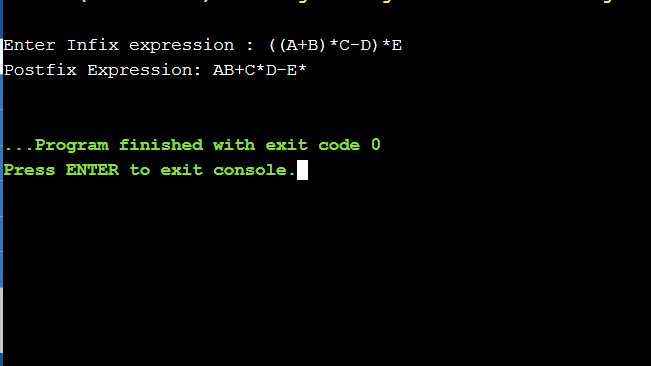
printf("Postfix Expression: ");

puts(postfix);

}

OUTPUT:

**(FOR A VALID INFIX EXPRESSION):**



**(FOR AN INVALID INFIX EXPRESSION):**

