#include <stdio.h>

#include<string.h>

#include<ctype.h>

#include<stdlib.h>

# define MAX 100

char stack[MAX];

int top=-1;

void push(char ch)

{

if (top==MAX-1)

printf("Stack is full\n");

else

{

top++;

stack[top]=ch;

}

}

char pop()

{

char item;

if (top==-1)

printf("\n stack is empty !");

else

{

item=stack[top];

top--;

return item;

}

}

int stackempty()

{

if(top==-1) return 1;

else return 0;

}

char stacktop()

{

if( top==-1)

printf("\n stack is empty!");

else

return stack[top];

}

int priority(char ch)

{

switch(ch)

{

case '+':

case '-':return (1);

case '\*':

case '/':return (2);

case '^': return (3);

default : return (0);

}

}

int main()

{

char infix[100],postfix[100];

int i,j, item;

printf("Enter the infix expression :");

scanf("%s",infix);

printf("Expression : %s",infix);

printf("\n Postfix: ");

i=0,j=0;

while (infix[i]!='\0')

{

switch (infix[i])

{

case '(': push(infix[i]);

break;

case ')':while(( item=pop())!='(')

{

postfix[j++]=item;

}

break;

case '+':

case '-':

case '\*':

case '/':

case '^':

while(!stackempty() && priority(infix[i])<=priority(stacktop()))

{

item=pop();

postfix[j++]=item;

}

push(infix[i]);

break;

default : if(isdigit(infix[i]) || isalpha(infix[i]))

{

postfix[j++]=infix[i];

}

else

{

printf("Invalid Infix Expression\n");

exit(1);

}

break;

}

i++;

}

while(!stackempty())

{

char item;

item=pop();

postfix[j++]=item;

}

postfix[j]='\0';

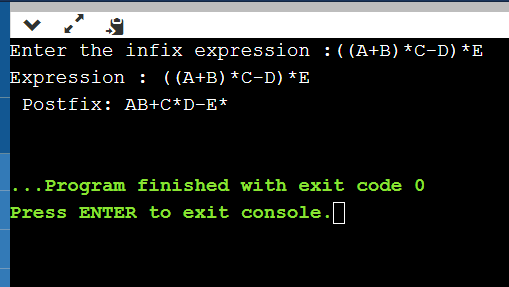
puts(postfix);

return 0;

}

**OUTPUT:**

**FOR A VALID INFIX EXPRESSION:**



**FOR AN INVALID INFIX EXPRESSION:**

