**Program for converting Infix expression to Postfix expression using Stacks.**

#include<stdio.h>

#include<ctype.h>

#define SIZE\_STACK 20

int STACK[SIZE\_STACK];

int TOP = 0;

int isSTACKFull(){

if (TOP == SIZE\_STACK)

return 1;

return 0;

}

void push(int item){

if (isSTACKFull()){

printf("Stack is Full. \n");

return;

}

STACK[TOP++] = item;

}

int isSTACKEmpty(){

if (TOP == 0)

return 1;

return 0;

}

int pop(){

if (isSTACKEmpty()){

printf("Stack is Empty. \n");

return -1;

}

return STACK[--TOP];

}

int getTopStack(){

if (isSTACKEmpty())

return -1; // if stack is empty.

return STACK[TOP - 1];

}

int getPrecedenceOfOperator(char c, int on\_stack){

switch (c){

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

case '^':

if (on\_stack)

return 9;

else

return 10;

case '(':

if (on\_stack)

return 0;

else

return 20;

default:

return -1; // if stack is empty.

}

}

void inFixToPostFix(char expre[]){

int i = 0;

while (expre[i] != '\0'){

char x = expre[i], tmp;

if ((x >= 65 && x <= 90) || (x >= 97 && x <= 122))

printf("%c", x);

else{

if (x == ')'){

while ((tmp = pop()) != '(')

printf("%c", tmp);

i++;

continue;

}

while (getPrecedenceOfOperator(x, 0) <=

getPrecedenceOfOperator(getTopStack(), 1))

printf("%c", pop());

push(x);

}

i++;

}

while (!isSTACKEmpty())

printf("%c", pop());

printf("\n");

}

int main(){

char infixExpre[25];

printf("Enter the infix expression: ");

gets(infixExpre);

inFixToPostFix(infixExpre);

return 0;

}

**Output:**

