Government College of Engineering, Karad Programming for Problem Solving Lab

Nanekar Saurabh Rajesh

20141212

I1

**Experiment No. 4**

**Title**: Implement stack as an ADT to perform expression conversion and evaluation for infix to prefix.

**Outcome:** Students can perform expression conversion and evaluation for infix to prefix and its related applications.

**Theory:**

While we use infix expressions in our day to day lives. Computers have trouble understanding this format because they need to keep in mind rules of operator precedence and also brackets. Prefix and Postfix expressions are easier for a computer to understand and evaluate.

Given two operands a and b and an operator •, the infix notation implies that O will be placed in between a and b i.e a • b . When the operator is placed after both operands i.e ab•, it is called postfix notation. And when the operator is placed before the operands i.e •a b, the expression in prefix notation.

Given any infix expression we can obtain the equivalent prefix and postfix format.

Input: A \* B + C / D

Output: + \* A B/ C D

Input: (A - B/C) \* (A/K-L)

Output: \*-A/BC-/AKL

To convert an infix to postfix expression refer to this article [Stack | Set 2 (Infix to Postfix)](https://www.geeksforgeeks.org/stack-set-2-infix-to-postfix/). We use the same to convert Infix to Prefix.

* Step 1: Reverse the infix expression i.e A+B\*C will become C\*B+A. Note while reversing each ‘(‘ will become ‘)’ and each ‘)’ becomes ‘(‘.
* Step 2: Obtain the postfix expression of the modified expression i.e CB\*A+.
* Step 3: Reverse the postfix expression. Hence in our example prefix is +A\*BC.

Algorithm to Convert Infix To Prefix:

* Let, X is an arithmetic expression written in infix notation. This algorithm finds the equivalent prefix expression Y.
* Reverse the infix expression.
* Make Every “ ( ” as “ ) ” and every “ ) ” as “ ( ”
* Push “ ( ” onto Stack, and add “ ) ” to the end of X.
* Scan X from left to right and repeat Step 3 to 6 for each element of X until the Stack is empty.
* If an operand is encountered, add it to Y.
* If a left parenthesis is encountered, push it onto Stack.
* If an operator is encountered ,then :
* Repeatedly pop from Stack and add to Y each operator (on the top of Stack) which has the same precedence as or higher precedence than operator.
* Add operator to Stack.  
  [ End of If ]
* If a right parenthesis is encountered ,then :
* Repeatedly pop from Stack and add to Y each operator (on the top of Stack) until a left parenthesis is encountered.
* Remove the left Parenthesis.  
  [ End of If ]  
  [ End of If ]
* Reverse the prefix expression.
* END.

**Example**

Infix Expression : A + ( B \* C - ( D / E ^ F ) \* G ) \* H

Reverse the infix expression :  
H \* ) G \* ) F ^ E / D ( - C \* B ( + A

Make Every “ ( ” as “ ) ” and every “ ) ” as “ ( ”  
H \* ( G \* ( F ^ E / D ) - C \* B ) + A

Convert expression to postfix form:

H \* ( G \* ( F ^ E / D ) - C \* B ) + A

|  |  |  |  |
| --- | --- | --- | --- |
| **Scanned** | **Stack** | **Postfix Expression** | **Description** |
|  | ( |  | Start |
| H | ( | H |  |
| \* | (\* | H |  |
| ( | (\*( | H |  |
| G | (\*( | HG |  |
| \* | (\*(\* | HG |  |
| ( | (\*(\*( | HG |  |
| F | (\*(\*( | HGF |  |
| ^ | (\*(\*(^ | HGF |  |
| E | (\*(\*(^ | HGFE |  |
| / | (\*(\*(/ | HGFE^ | ' ^ ' is at highest precedence then ' / ' |
| D | (\*(\*(/ | HGFE^D |  |
| ) | (\*(\* | HGFE^D/ |  |
| - | (\*(- | HGFE^D/∗ |  |
| C | (\*(- | HGFE^D/∗C |  |
| \* | (\*(-\* | HGFE^D/∗C |  |
| B | (\*(-\* | HGFE^D/∗CB |  |
| ) | (\* | HGFE^D/∗CB∗- | POP from top on Stack, that's why ' \* ' come first |
| + | (+ | HGFE^D/∗CB∗-∗ | ' \* ' is at highest precedence then ' + ' |
| A | (+ | HGFE^D/∗CB∗-∗A |  |
| ) | Empty | HGFE^D/∗CB∗-∗A+ | END |

Reverse the postfix expression:   
+A\*-\*BC\*/D^EFGH

**Complexity:**   
Stack operations like push() and pop() are performed in constant time. Since we scan all the characters in the expression once the complexity is linear in time i.e .o(n)

**Evaluation of Prefix Expressions**

Prefix and Postfix expressions can be evaluated faster than an infix expression. This is because we don’t need to process any brackets or follow operator precedence rule. In postfix and prefix expressions which ever operator comes before will be evaluated first, irrespective of its priority. Also, there are no brackets in these expressions. As long as we can guarantee that a valid prefix or postfix expression is used, it can be evaluated with correctness.

**Algorithm**

EVALUATE\_PREFIX (STRING)

Step 1: Put a pointer P at the end of the end

Step 2: If character at P is an operand push it to Stack

Step 3: If the character at P is an operator pop two

elements from the Stack. Operate on these elements

according to the operator, and push the result

back to the Stack

Step 4: Decrement P by 1 and go to Step 2 as long as there

are characters left to be scanned in the expression.

Step 5: The Result is stored at the top of the Stack,

return it

Step 6: End

Expression: +9\*26

Character | Stack | Explanation

Scanned | (Front to |

| Back) |

-------------------------------------------

6 6 6 is an operand,

push to Stack

2 6 2 2 is an operand,

push to Stack

\* 12 (6\*2) \* is an operator,

pop 6 and 2, multiply

them and push result

to Stack

9 12 9 9 is an operand, push

to Stack

+ 21 (12+9) + is an operator, pop

12 and 9 add them and

push result to Stack

Result: 21

**Examples:**

Input : -+8/632

Output :

Input : -+7\*45+20

Output :

**Complexity:** The algorithm has linear complexity since we scan the expression once and perform at most O (N) push and pop operations which take constant time.

**Analysis:**



**List of similar programs: Solve any one.**

1. Write a program for implementation of [Infix to prefix conversion using two stacks.](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)
2. Write a program to print all words matching a pattern in CamelCase Notation Dictionary.

**Title Program:**Implement stack as an ADT to perform expression conversion and evaluation for infix to prefix.

**Source code of Implemented Programs:**

//Nanekar Saurabh Rajesh

#include<stdio.h>

#include<math.h>

#include<string.h>

#include <stdlib.h>

#define MAX 20

void push(int);

char pop();

void infix\_to\_prefix();

int precedence (char);

char stack[20],infix[20],prefix[20];

int top = -1;

int main()

{

printf("\nINPUT THE INFIX EXPRESSION : ");

scanf("%s",infix);

infix\_to\_prefix();

return 0;

}

void push(int pos)

{

if(top == MAX-1)

{

printf("\nSTACK OVERFLOW\n");

}

else {

top++;

stack[top] = infix[pos];

}}

char pop()

{

char ch;

if(top < 0)

{

printf("\nSTACK UNDERFLOW\n");

exit(0);

}

else

{

ch = stack[top];

stack[top] = '\0';

top--;

return(ch);

}

return 0;

}

void infix\_to\_prefix()

{

int i = 0,j = 0;

strrev(infix);

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

{

if(infix[i] >= 'a' && infix[i] <= 'z')

{

prefix[j] = infix[i];

j++;

i++;

}

else if(infix[i] == ')' || infix[i] == '}' || infix[i] == ']')

{

push(i);

i++;

}

else if(infix[i] == '(' || infix[i] == '{' || infix[i] == '[')

{

if(infix[i] == '(')

{

while(stack[top] != ')')

{

prefix[j] = pop();

j++;

}

pop();

i++;

}

else if(infix[i] == '[')

{

while(stack[top] != ']')

{

prefix[j] = pop();

j++;

}

pop();

i++;

}

else if(infix[i] == '{')

{

while(stack[top] != 0)

{

prefix[j] = pop();

j++;

}

pop();

i++;

}}

else

{

if(top == -1)

{

push(i);

i++;

}

else if( precedence(infix[i]) < precedence(stack[top]))

{

prefix[j] = pop();

j++;

while(precedence(stack[top]) > precedence(infix[i])){

prefix[j] = pop();

j++;

if(top < 0) {

break;

}}

push(i);

i++;

}

else if(precedence(infix[i]) >= precedence(stack[top]))

{

push(i);

i++;

}}}

while(top != -1)

{

prefix[j] = pop();

j++;

}

strrev(prefix);

prefix[j] = '\0';

printf("EQUIVALENT PREFIX NOTATION : %s ",prefix);

}

int precedence(char alpha)

{

if(alpha == '+' || alpha =='-')

{

return(1);

}

if(alpha == '\*' || alpha =='/')

{

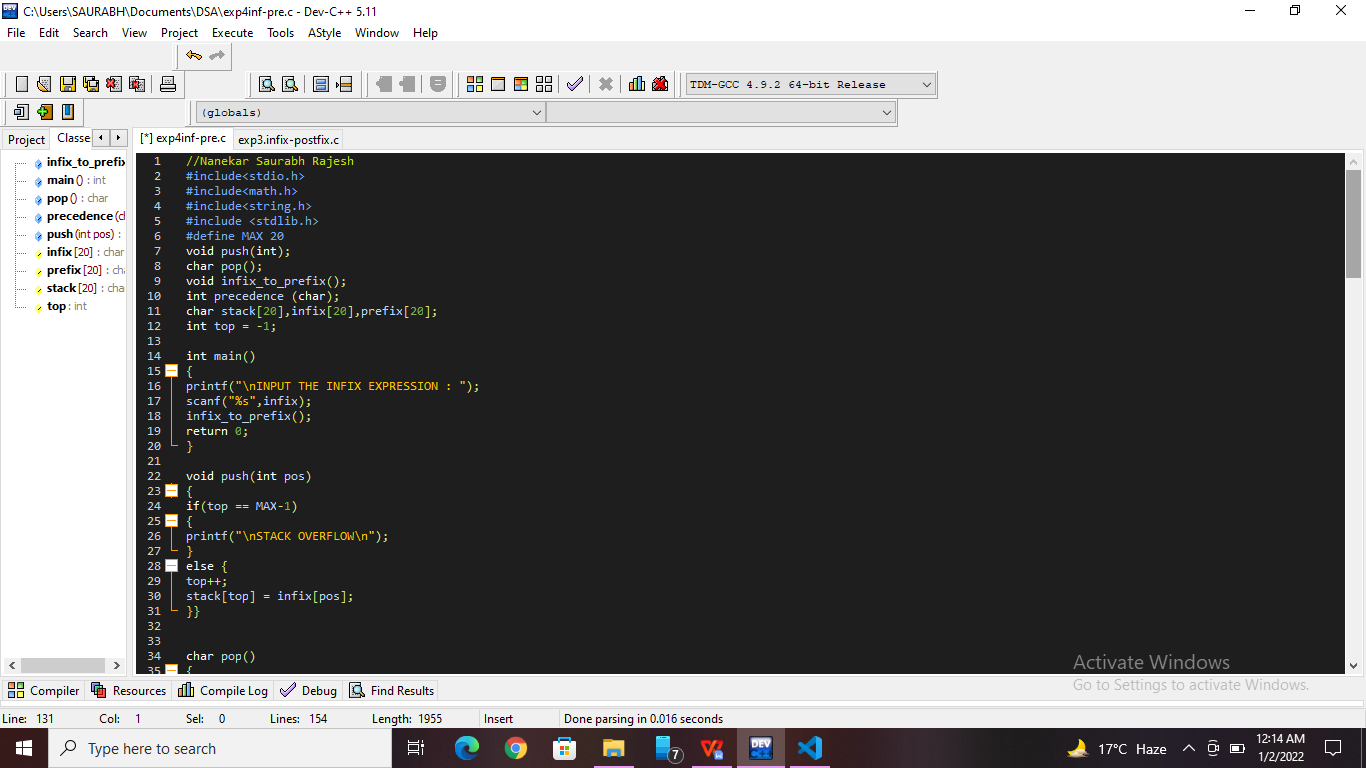
return(2);

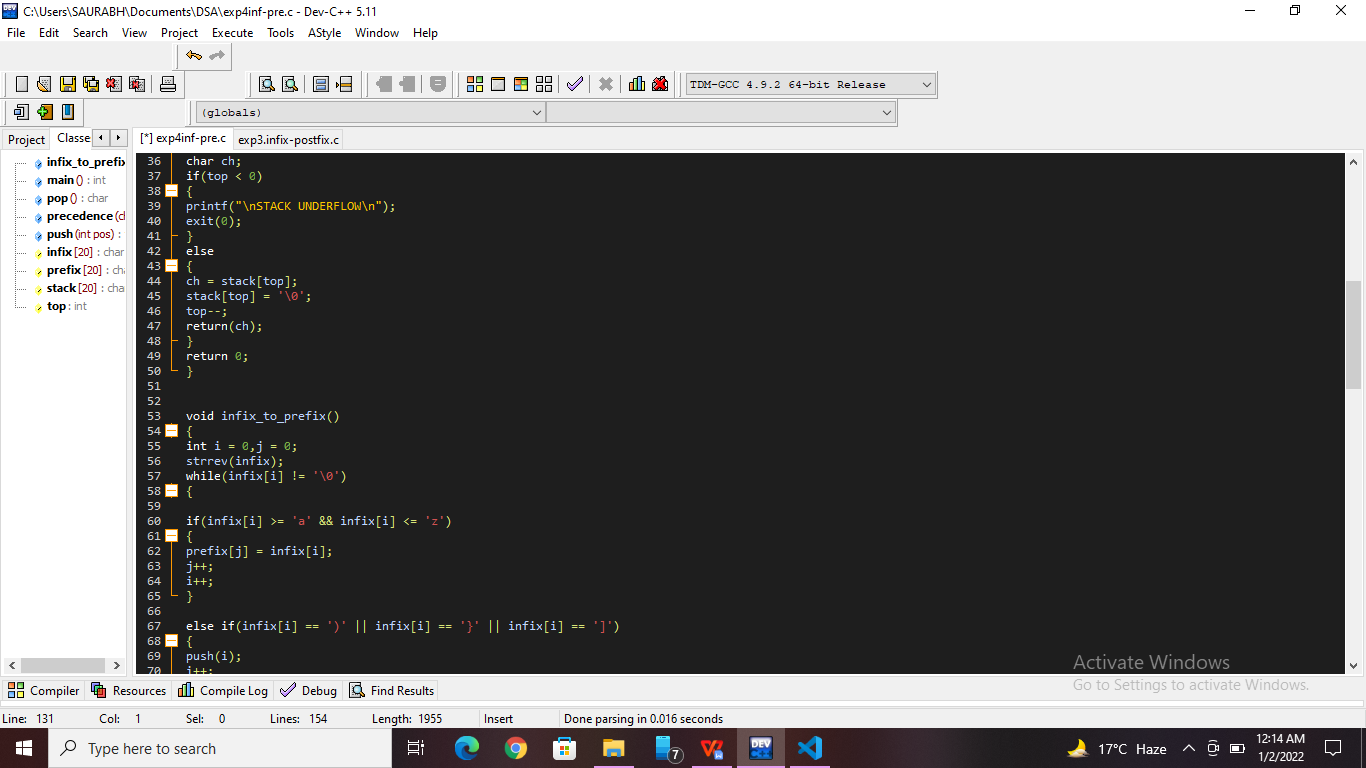
}

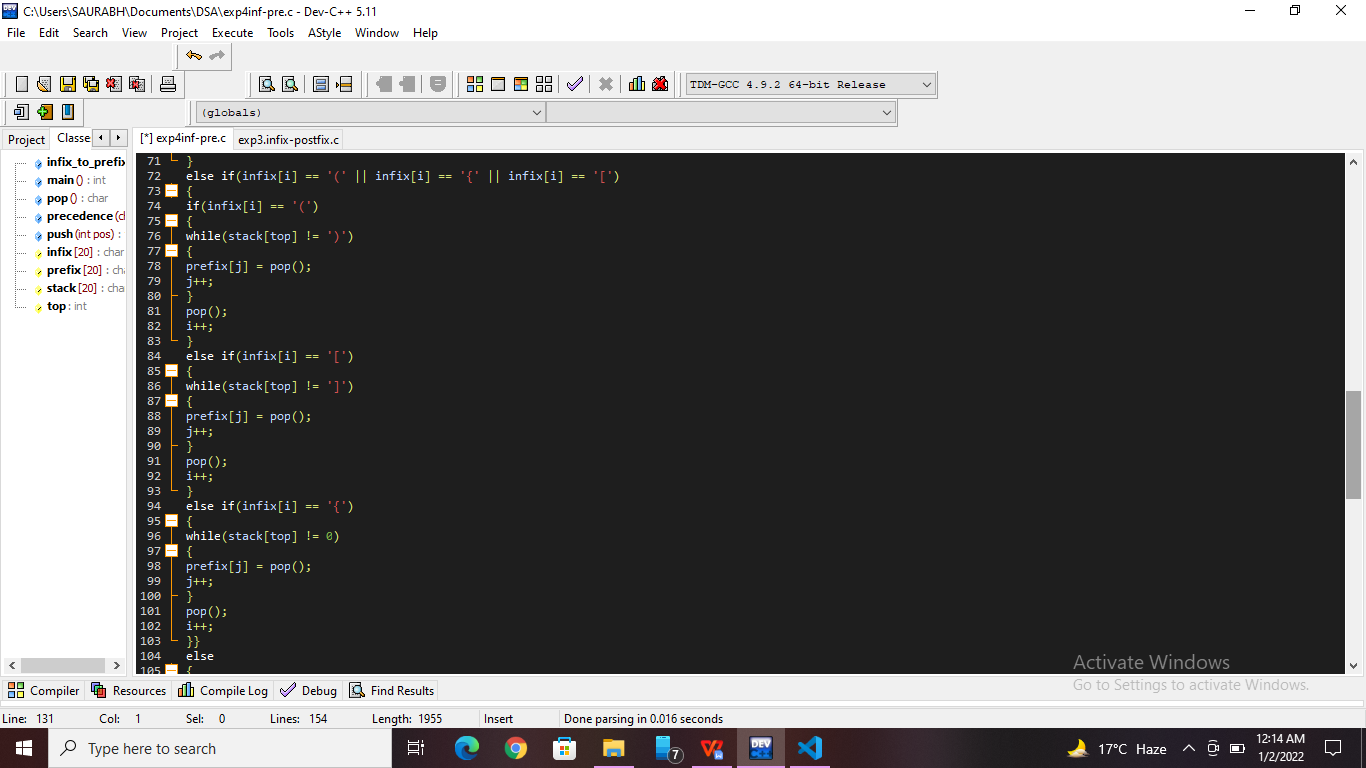
return 0;

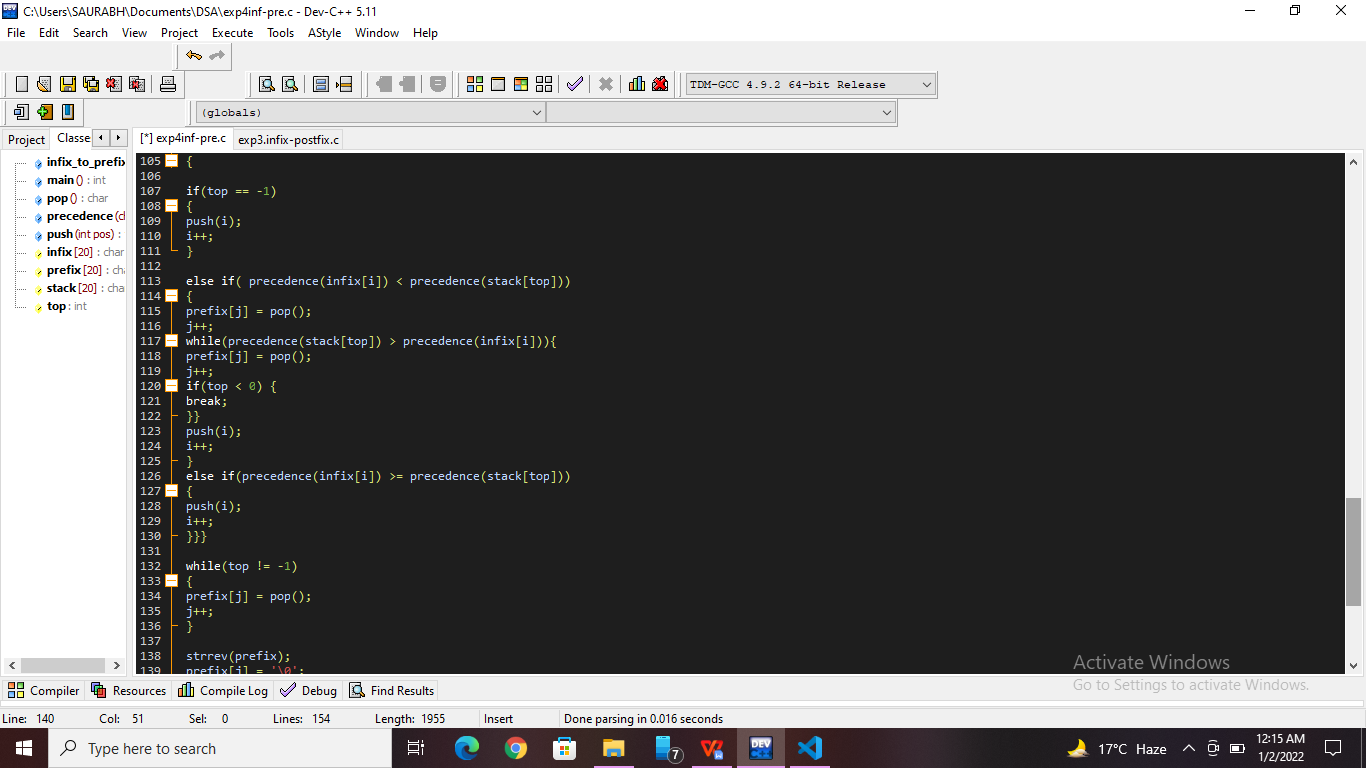
}

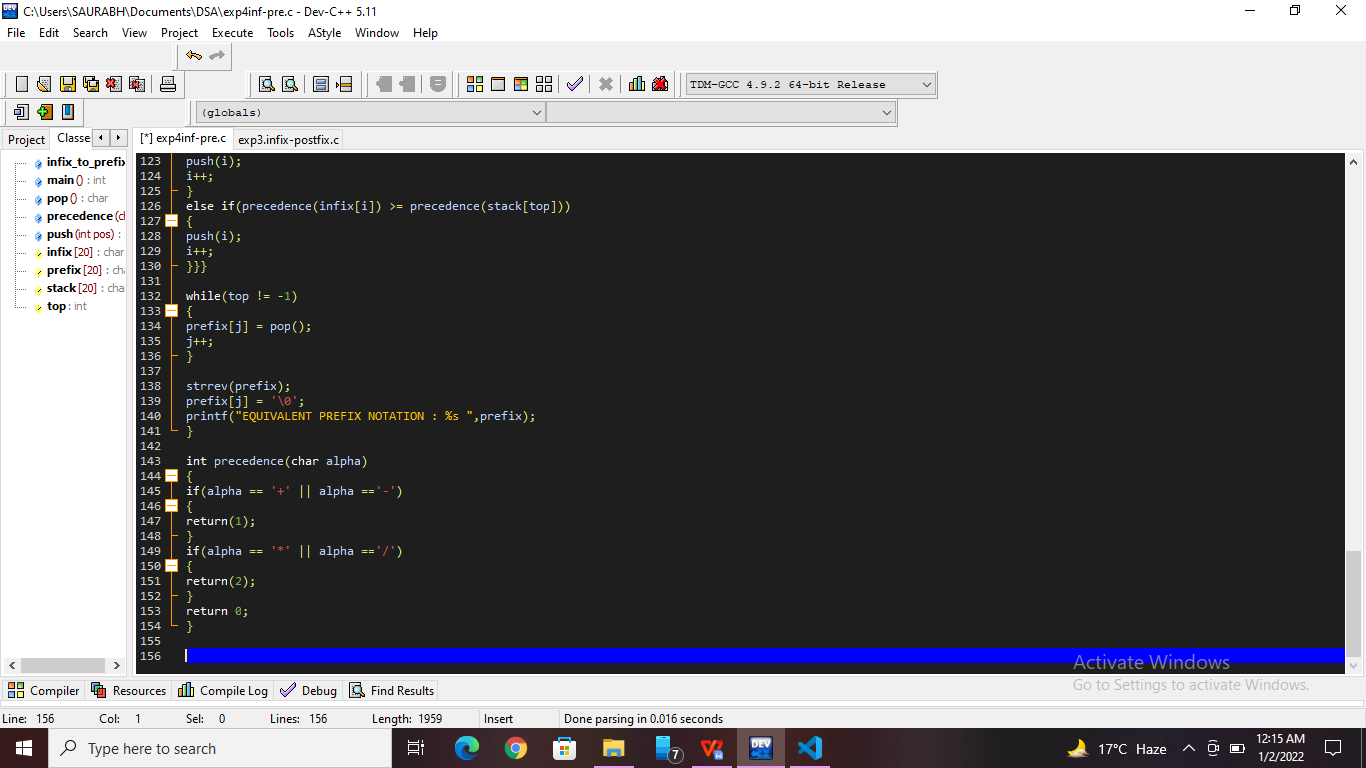
**Screenshots of Output:**

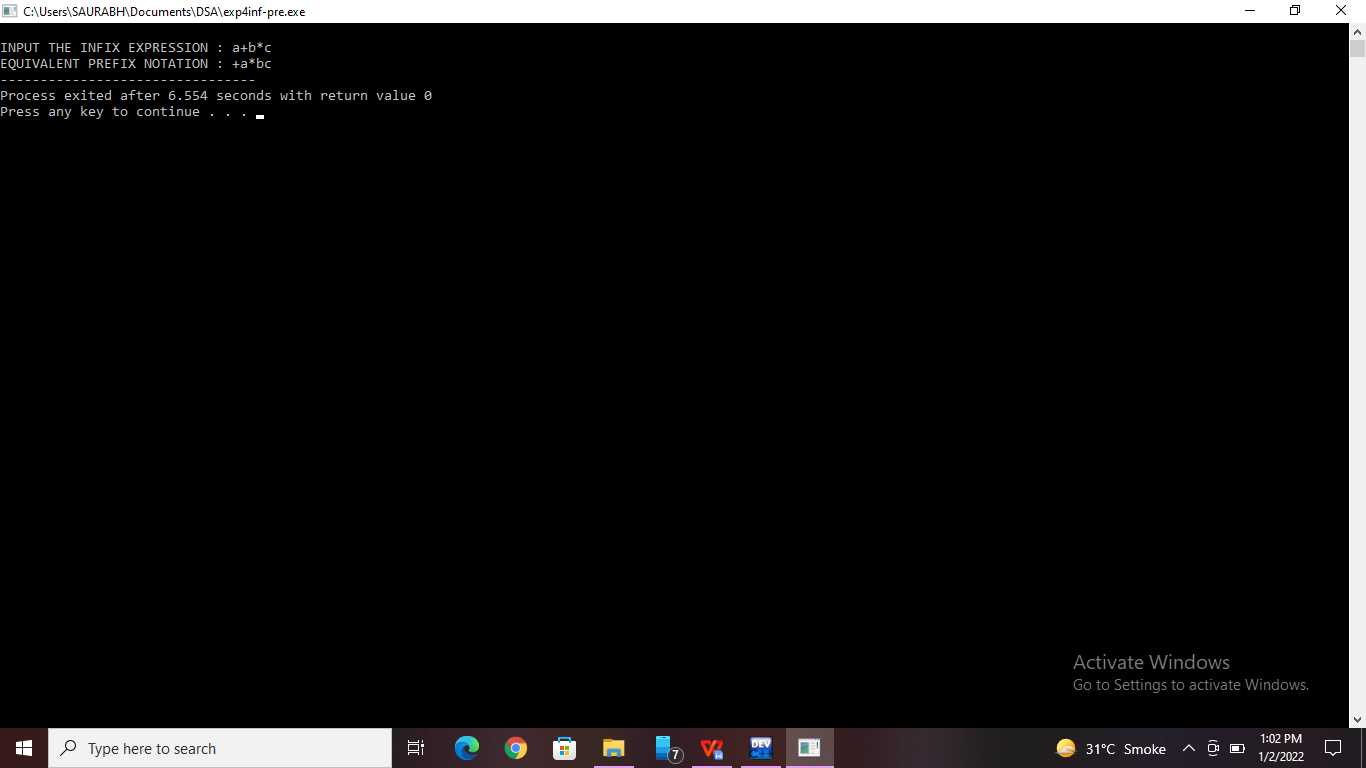
****

****

****

****

****

****

**Practice Program:** Write a program for implementation of [Infix to prefix conversion using two stacks.](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[//Nanekar Saurabh rajesh\_20141212\_I1](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[#include<stdio.h>](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[#include<conio.h>](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[#include<string.h>](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[#include<stdbool.h>](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[#define MAX 100](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[char s1[MAX][MAX], s2[MAX];](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[int top1=-1,top2=-1;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[void push1(char \*s)](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[if(top1>=MAX-1)](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[strcpy(s1[++top1],s);](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[char \*pop1()](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return s1[top1--];](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[char \*Top1()](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return s1[top1];](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[bool empty1()](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[if(top1==-1)](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return true;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[else](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return false;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[void push2(char c)](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[if(top1>=MAX-1)](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[s2[++top2]=c;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[char pop2()](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return s2[top2--];](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[char Top2()](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return s2[top2];](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[bool empty2()](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[if(top2==-1)](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return true;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[else](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return false;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[bool isOperator(char c)](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[if(c=='\*' || c=='/' || c=='+' || c=='-' || c=='^')](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return true;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[else](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return false;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[bool isOperand(char c)](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[if(c>='a' && c<='z' || c>='A' || c<='Z')](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return true;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[else](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return true;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[int getPriority(char c)](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[if(c=='\*' || c=='/' || c=='%')](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return 1;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[else if(c=='+' || c=='-')](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return 0;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[else if(c=='^')](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return 2;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[else](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[return -1;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[void process()](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[char a[50],b[50],c[2];](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[c[0]=pop2();](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[c[1]='\0';](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[strcpy(a,c);](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[strcat(a,pop1());](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[strcat(a,pop1());](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[push1(a);](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[void infixToPrefix(char \*s,int n)](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[int i;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[char res[n];](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[for(i=n-1;i>=0;i--)](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[if(s[i]=='(')](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[push2(s[i]);](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[else if(isOperator(s[i]))](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[while(!empty2() && getPriority(s[i])<getPriority(Top2()))](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[process();](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[push2(s[i]);](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[else if(isOperand(s[i]))](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[char a[2];](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[a[0]=s[i];](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[a[1]='\0';](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[push1(a);](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[else if(s[i]==')')](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[while(!empty2() && Top2()!='(')](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[process();](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[pop2();](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[while(!empty2())](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[process();](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[strcpy(res,pop1());](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[printf("\nPrefix Expression: ");](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[puts(res);](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[int main()](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[{](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[char str[100];](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[printf("Enter an infix string: ");](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

[gets(str);](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

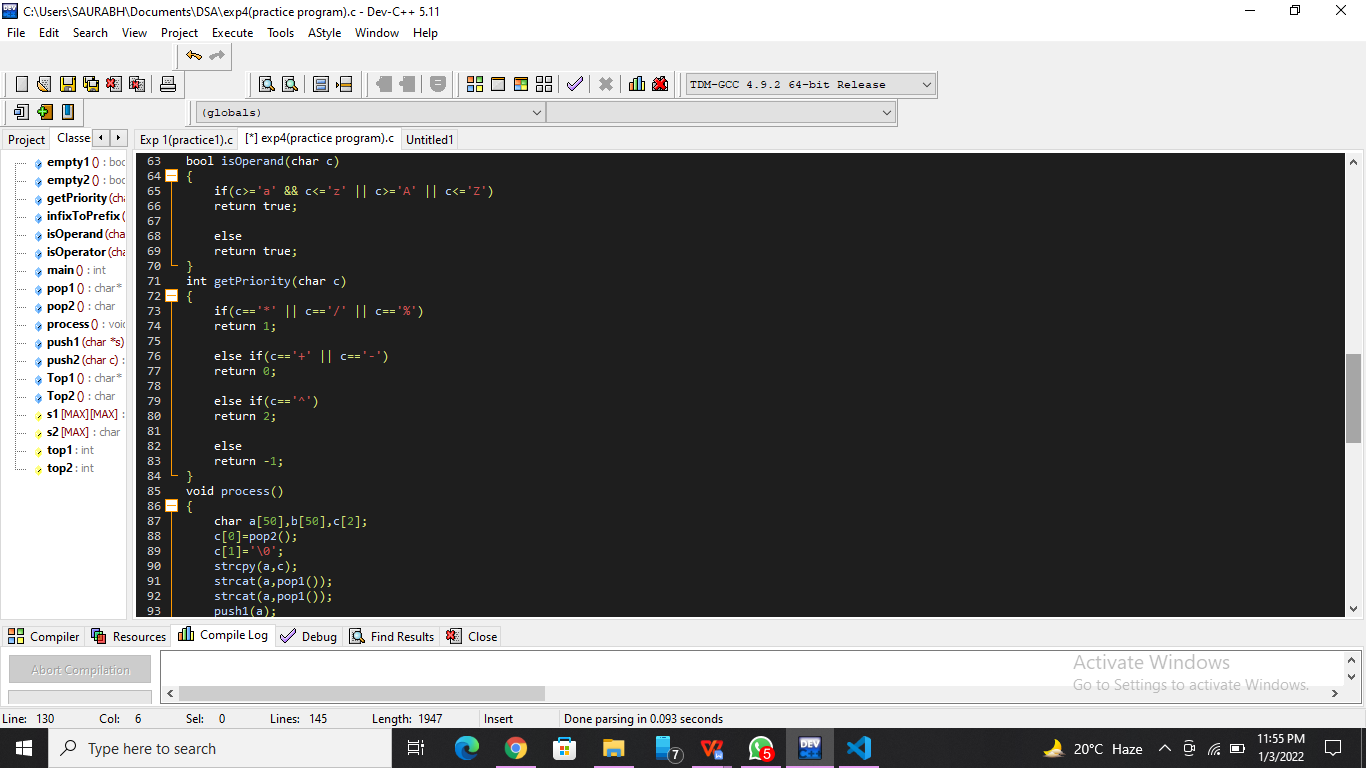
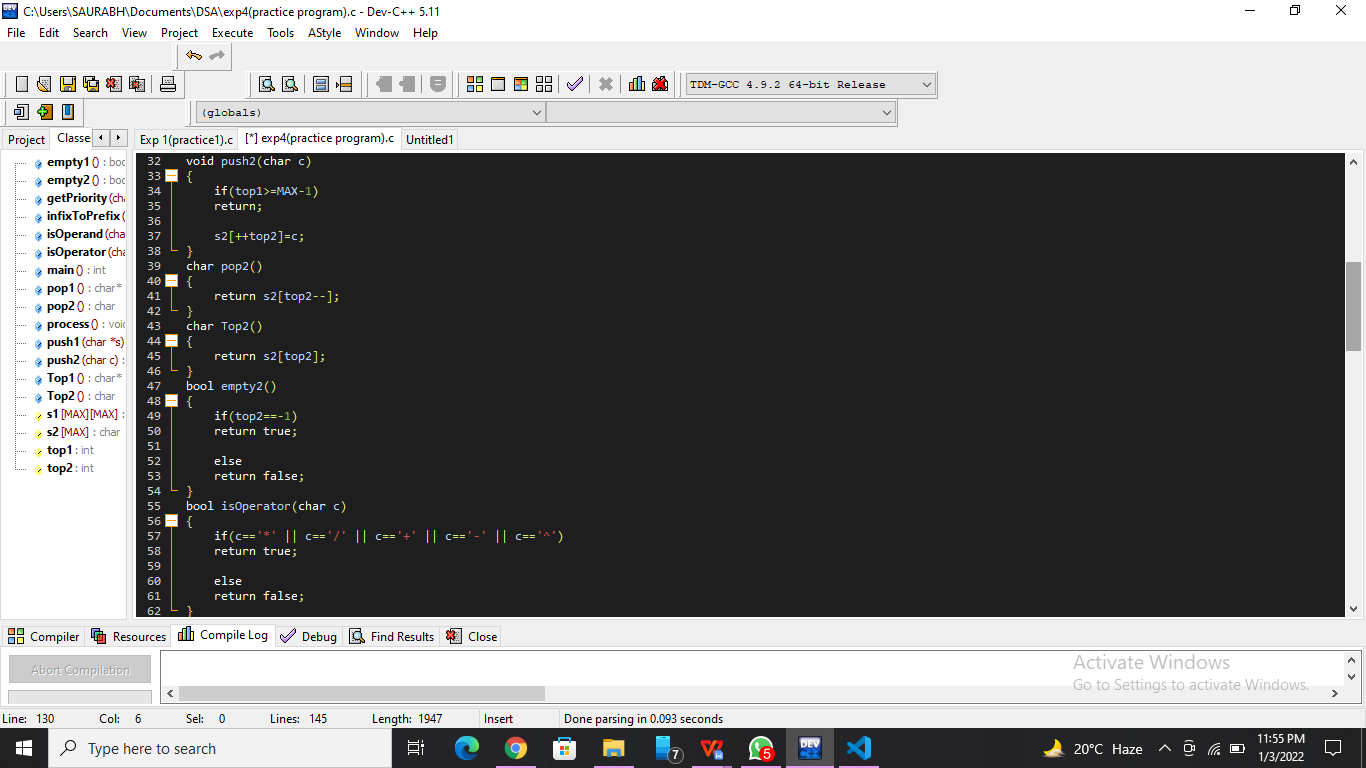
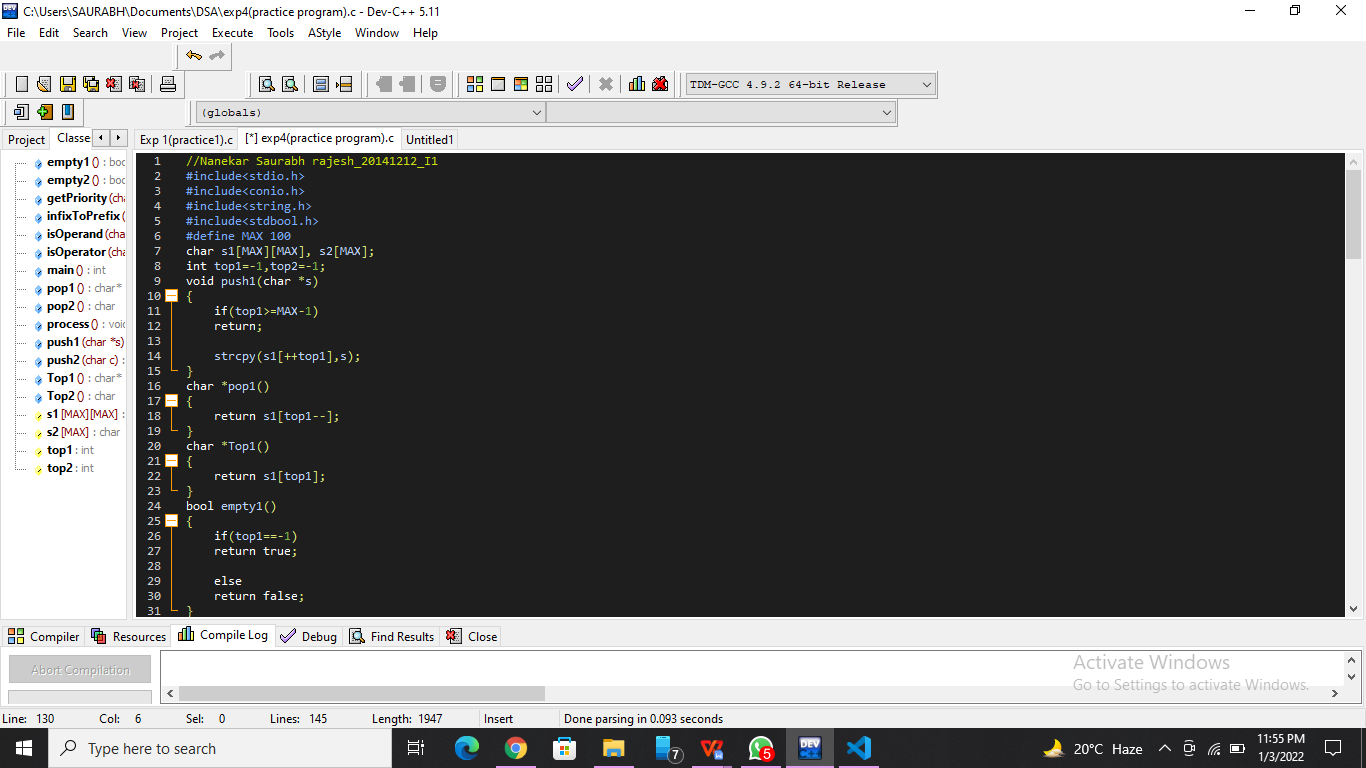
[int n=strlen(str);](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

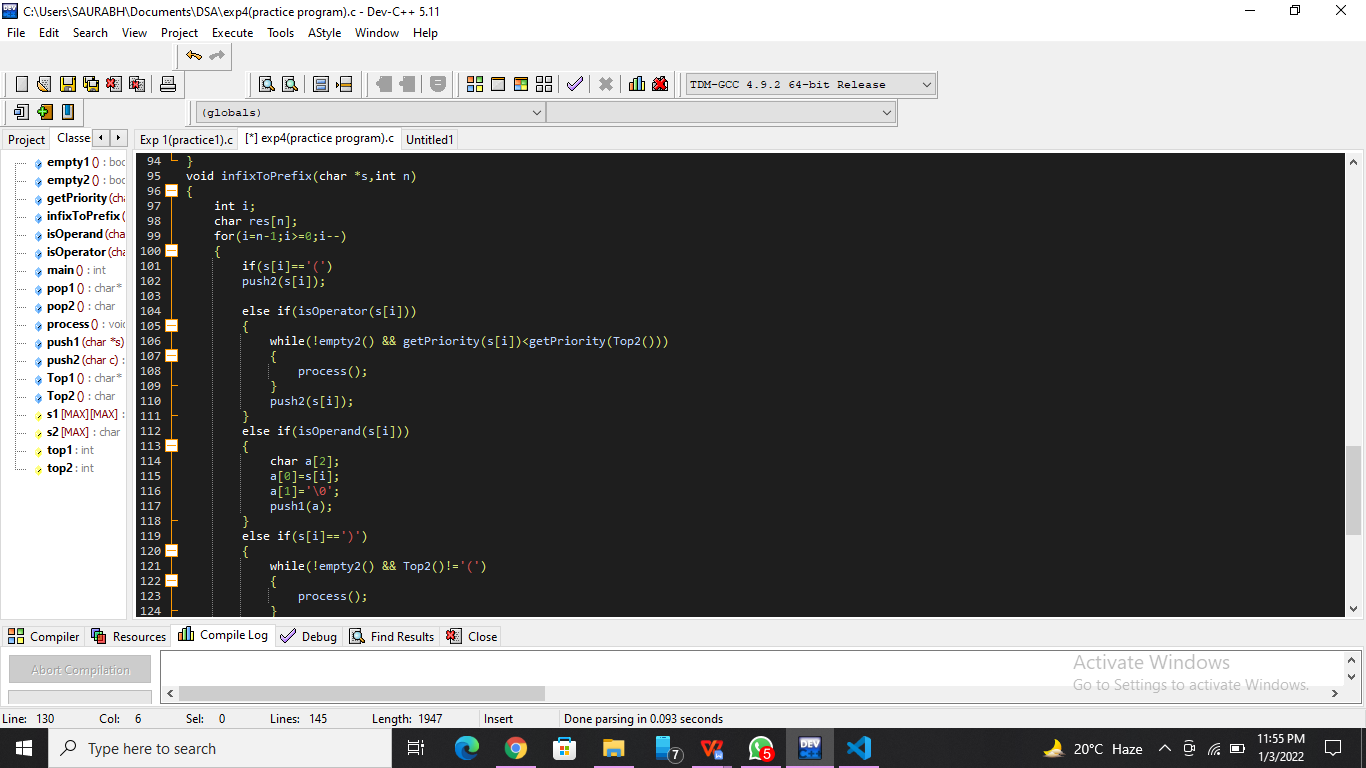
[infixToPrefix(str,n);](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

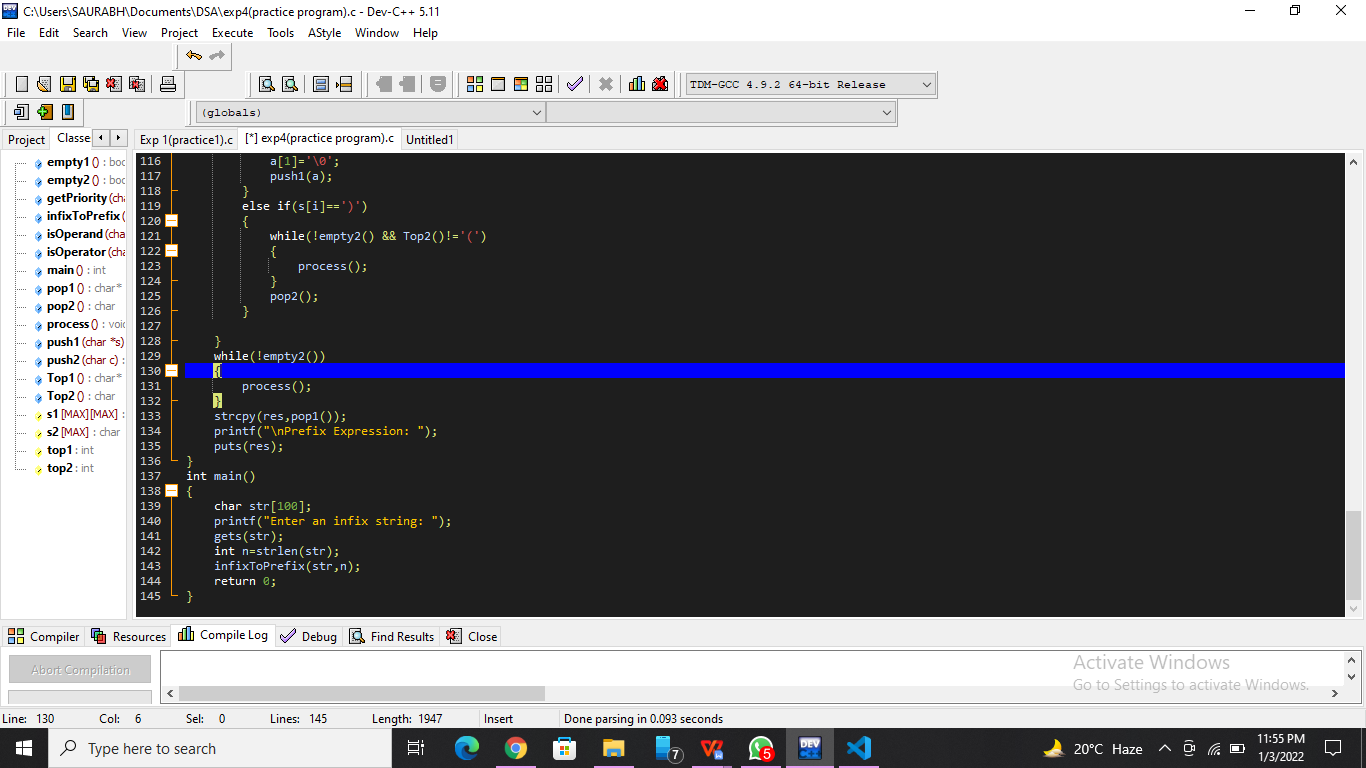
[return 0;](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

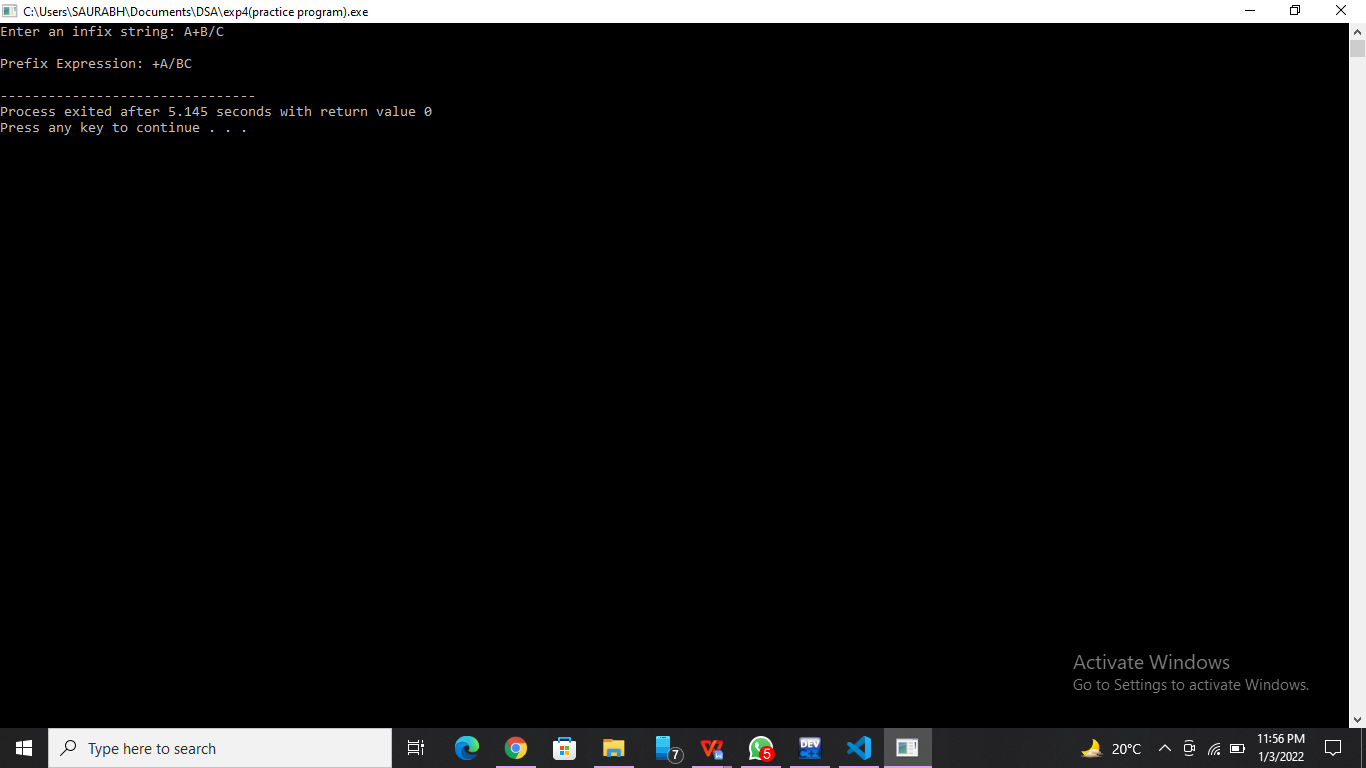
[}](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)

**[Screenshots of Practice Program:-](https://www.geeksforgeeks.org/infix-to-prefix-conversion-using-two-stacks/?ref=rp" \o "Permalink to Infix to Prefix conversion using two stacks)**









**List of sample questions for oral examination:**

1. Which will you use to convert infix to postfix prefix operators?
2. How do I convert infix to prefix manually?
3. Which is required to convert the infix to prefix notation?
4. How do you solve an infix expression?
5. What is infix prefix and postfix?

**Conclusion:**

We can easily distinguish the order of operators, and also can use the parenthesis to solve that part first during solving mathematical expressions.