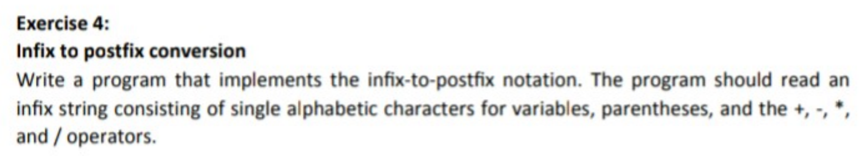
Question: Exercise 4: Infix to postfix conversion Write a program that im...



Exercise 4:

Infix to postfix conversion

Write a program that implements the infix-to-postfix notation. The program should read an infix string consisting of single alphabetic characters for variables, parentheses, and the +, -, *, and / operators.

write the program in c++ language

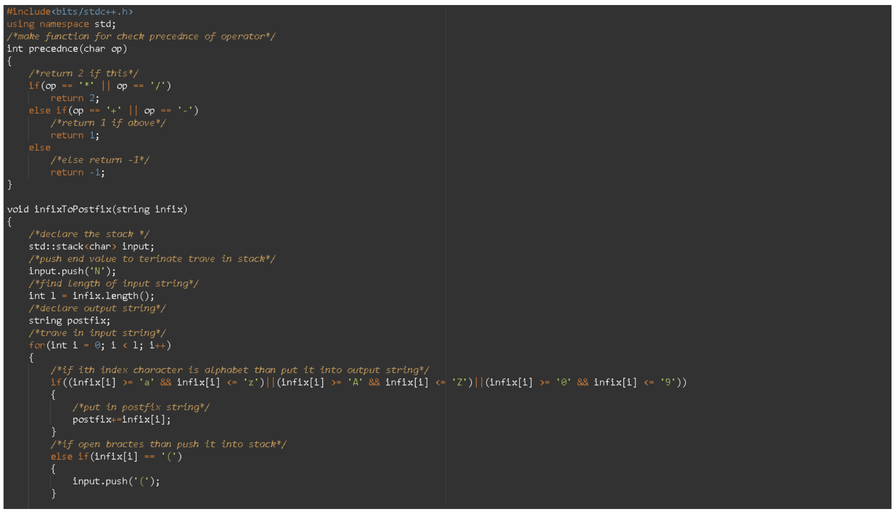
Show transcribed image text

Expert Answer

Anonymous answered this
1,614 answers

Was this answer helpful? 1 0

input code:



```
#include<bits/stdc++.h>
using namespace std;
/*make function for check precednce of operator*/
int precednce(char op)
{
    /*return 2 if this*/
    if(op == '*' || op == '/')
        return 2;
    else if(op == '+' || op == '-')
        /*return 1 if above*/
        return 1;
    else
        /*else return -1*/
        return -1;
}

void infixToPostfix(string infix)
{
    /*declare the stack */
    std::stack<char> input;
    /*push end value to terinate trave in stack*/
    input.push('N');
    /*find length of input string*/
    int l = infix.length();
    /*declare output string*/
    string postfix;
    /*trave in input string*/
    for(int i = 0; i < l; i++)
    {
        /*if ith index character is alphabet than put it into output string*/
        if((infix[i] >= 'a' && infix[i] <= 'z')||(infix[i] >= 'A' && infix[i] <= 'Z')||(infix[i] >= '0' && infix[i] <= '9'))
        {
            /*put in postfix string*/
            postfix+=infix[i];
        }
        /*if open bractes than push it into stack*/
        else if(infix[i] == '(')
        {
            input.push('(');
        }
```

Post a question

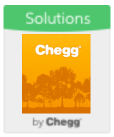
Answers from our experts for your t homework questions

Enter question

Continue to pos

0 questions remaining

My Textbook Solutions

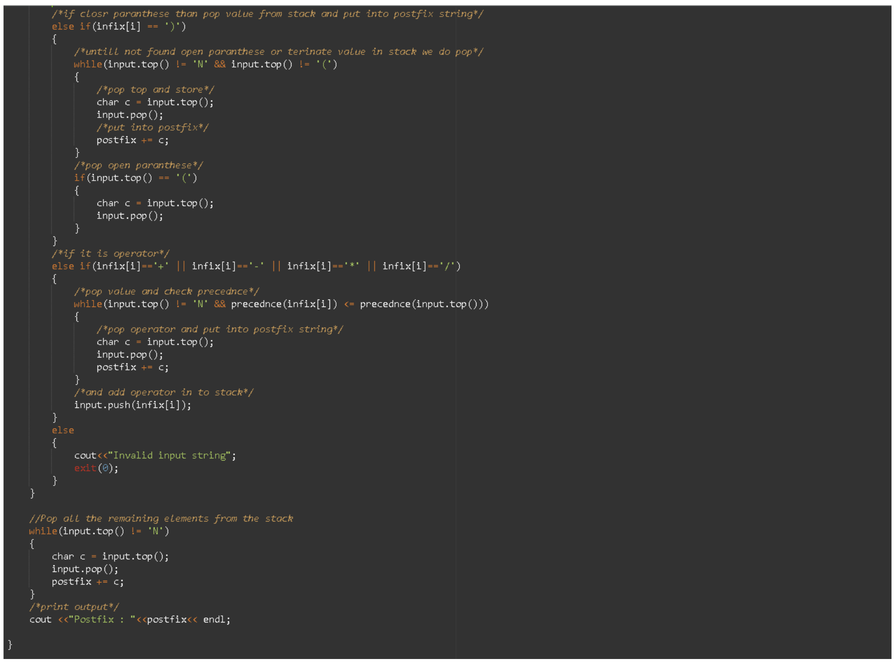


International Accounting 5th Edition

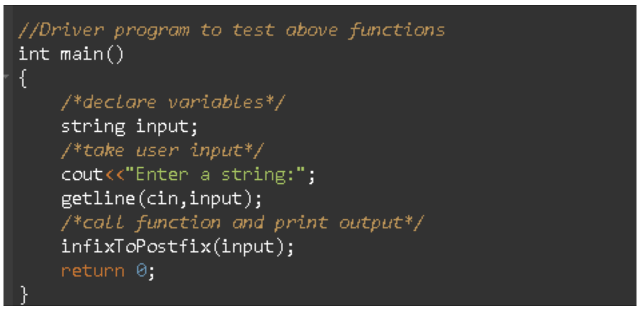
Principles of Electronic... 3rd Edition

In to 5th

View all solutions

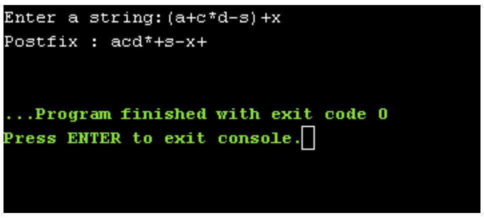


```
        /*if closr paranthese than pop value from stack and put into postfix string*/
        else if(infix[i] == ')')
        {
            /*untill not found open paranthese or terinate value in stack we do pop*/
            while(input.top() != 'N' && input.top() != '(')
            {
                /*pop top and store*/
                char c = input.top();
                input.pop();
                /*put into postfix*/
                postfix += c;
            }
            /*pop open paranthese*/
            if(input.top() == '(')
            {
                char c = input.top();
                input.pop();
            }
        }
        /*if it is operator*/
        else if(infix[i]=='+' || infix[i]=='-' || infix[i]=='*' || infix[i]=='/')
        {
            /*pop value and check precednce*/
            while(input.top() != 'N' && precednce(infix[i]) <= precednce(input.top()))
            {
                /*pop operator and put into postfix string*/
                char c = input.top();
                input.pop();
                postfix += c;
            }
            /*and add operator in to stack*/
            input.push(infix[i]);
        }
        else
        {
            cout<<"Invalid input string";
            exit(0);
        }
    }

    //Pop all the remaining elements from the stack
    while(input.top() != 'N')
    {
        char c = input.top();
        input.pop();
        postfix += c;
    }
    /*print output*/
    cout <<"Postfix : "<<postfix<< endl;

}
```



```
//Driver program to test above functions
int main()
{
    /*declare variables*/
    string input;
    /*take user input*/
    cout<<"Enter a string:";
    getline(cin,input);
    /*call function and print output*/
    infixToPostfix(input);
    return 0;
}
```

output:



```
Enter a string:(a+c*d-s)+x
Postfix : acd*+s-x+


...Program finished with exit code 0
Press ENTER to exit console.
```

code:

```
#include<bits/stdc++.h>
using namespace std;
/*make function for check precednce of operator*/
int precednce(char op)
{
/*return 2 if this*/
if(op == '*' || op == '/')
return 2;
else if(op == '+' || op == '-')
/*return 1 if above*/
return 1;
else
/*else return -1*/
return -1;
}

void infixToPostfix(string infix)
{
/*declare the stack */
std::stack<char> input;
/*push end value to terinate trave in stack*/
input.push('N');
/*find length of input string*/
int l = infix.length();
/*declare output string*/
```

```
string postfix;
/*trave in input string*/
for(int i = 0; i < l; i++)
{
/*if ith index character is alphabet than put it into output string*/
if((infix[i] >= 'a' && infix[i] <= 'z')||(infix[i] >= 'A' && infix[i] <= 'Z')||(infix[i] >= '0' && infix[i] <= '9'))
{
/*put in postfix string*/
postfix+=infix[i];
}
/*if open bractes than push it into stack*/
else if(infix[i] == '(')
{
input.push('(');
}


/*if closr paranthese than pop value from stack and put into postfix string*/
else if(infix[i] == ')')
{
/*untill not found open paranthese or terinate value in stack we do pop*/
while(input.top() != 'N' && input.top() != '(')
{
/*pop top and store*/
char c = input.top();
input.pop();
/*put into postfix*/
postfix += c;
}
/*pop open paranthese*/
if(input.top() == '(')
{
char c = input.top();
input.pop();
}
}
/*if it is operator*/
else if(infix[i]=='+' || infix[i]=='-' || infix[i]=='*' || infix[i]=='/')
{
/*pop value and check precednce*/
while(input.top() != 'N' && precednce(infix[i]) <= precednce(input.top()))
{
/*pop operator and put into postfix string*/
char c = input.top();
input.pop();
postfix += c;
}
/*and add operator in to stack*/
input.push(infix[i]);
}
else
{
cout<<"Invalid input string";
exit(0);
}
}

//Pop all the remaining elements from the stack
while(input.top() != 'N')
{
char c = input.top();
input.pop();
postfix += c;
}
/*print output*/
cout <<"Postfix : "<<postfix<< endl;

}

//Driver program to test above functions
int main()
{
/*declare variables*/
string input;
/*take user input*/
cout<<"Enter a string:";
getline(cin,input);
/*call function and print output*/
infixToPostfix(input);
return 0;
```

}

Comment >

COMPANY

About Chegg
Chegg For Good
College Marketing
Corporate Development
Investor Relations
Jobs
Join Our Affiliate Program
Media Center
Site Map

LEGAL & POLICIES

Advertising Choices
Cookie Notice
General Policies
Intellectual Property Rights
Terms of Use
Global Privacy Policy
DO NOT SELL MY INFO
Honor Code
Honor Shield

CHEGG PRODUCTS AND SERVICES

Cheap Textbooks
Chegg Coupon
Chegg Play
Chegg Study Help
College Textbooks
eTextbooks
Flashcards
Learn
Uversity
Chegg Math Solver
Mobile Apps
Sell Textbooks
Solutions Manual
Study 101
Textbook Rental
Used Textbooks
Digital Access Codes
Chegg Life
Chegg Writing

CHEGG NETWORK

EasyBib
Internships.com
Thinkful

CUSTOMER SERVICE

Customer Service
Give Us Feedback
Manage Subscription

© 2003-2022 Chegg Inc. All rights reserved.