NAME: Muhammad Haadhee Sheeraz Mian

Reg no: 478359

**Assignment #1**

**Write C++ program to convert infix expression into postfix and prefix conversions and from postfix and prefix to infix expression. Also calculate the time complexity of these functions. using infix expression.**

• **Upload source code**

• **One document for output screen shot and highlight in which function you have taken help from chatgpt.**

**Code:**

#include <iostream>

#include <stack>

#include <algorithm>

using namespace std;

// Function to check operator precedence

int precedence(char op) {

    if(op == '+' || op == '-') return 1;

    if(op == '\*' || op == '/') return 2;

    return 0;

}

// Check if character is operator

bool isOperator(char c) {

    return (c == '+' || c == '-' || c == '\*' || c == '/');

}

// Convert infix to postfix

string infixToPostfix(string infix) {

    stack<char> st;

    string postfix = "";

    for(char c : infix) {

        if(isalnum(c)) {

            postfix += c;

        } else if(isOperator(c)) {

            while(!st.empty() && precedence(st.top()) >= precedence(c)) {

                postfix += st.top();

                st.pop();

            }

            st.push(c);

        }

    }

    while(!st.empty()) {

        postfix += st.top();

        st.pop();

    }

    return postfix;

}

// Convert infix to prefix

string infixToPrefix(string infix) {

    reverse(infix.begin(), infix.end());

    for(char& c : infix) {

        if(c == '(') c = ')';

        else if(c == ')') c = '(';

    }

    string postfix = infixToPostfix(infix);

    reverse(postfix.begin(), postfix.end());

    return postfix;

}

// Convert postfix to infix

string postfixToInfix(string postfix) {

    stack<string> st;

    for(char c : postfix) {

        if(isalnum(c)) {

            st.push(string(1, c));

        } else if(isOperator(c)) {

            string op2 = st.top(); st.pop();

            string op1 = st.top(); st.pop();

            st.push("(" + op1 + c + op2 + ")");

        }

    }

    return st.top();

}

// Convert prefix to infix

string prefixToInfix(string prefix) {

    stack<string> st;

    reverse(prefix.begin(), prefix.end());

    for(char c : prefix) {

        if(isalnum(c)) {

            st.push(string(1, c));

        } else if(isOperator(c)) {

            string op1 = st.top(); st.pop();

            string op2 = st.top(); st.pop();

            st.push("(" + op1 + c + op2 + ")");

        }

    }

    return st.top();

}

// Driver code

int main() {

    string infix = "A+B\*C";

    cout << "Infix: " << infix << endl;

    string postfix = infixToPostfix(infix);

    cout << "Postfix: " << postfix << endl;

    string prefix = infixToPrefix(infix);

    cout << "Prefix: " << prefix << endl;

    cout << "Postfix to Infix: " << postfixToInfix(postfix) << endl;

    cout << "Prefix to Infix: " << prefixToInfix(prefix) << endl;

    return 0;

}

Output:

A screen shot of a computer

AI-generated content may be incorrect.

**Time Complexity:**

* Infix to Postfix: O(n)
* Infix to Prefix: O(n) (because it uses O(n) reversal + O(n) infix→postfix)
* Postfix to Infix: O(n)
* Prefix to Infix: O(n)