Assignment 2

Data Structures

Roll no :20p-0480

Q1: Write a function to evaluate a postfix expression using a stack. The function should take a

postfix expression as a parameter and return the result of the expression.

Code:

#include <iostream>

#include <stack>

#include <string>

#include <cstdlib>

using namespace std;

int expression(string exp) {

    // Create a stack for operands

    stack<int> s;

    // use loop for each character of the expression

    for(char c : exp) {

        //for push in stack

        if(isdigit(c)) {

            s.push(c - '0');

        } else {

            int op2 = s.top();

            s.pop();

            int op1 = s.top();

            s.pop();

            switch(c) {

                case '+':

                    s.push(op1 + op2);

                    break;

                case '-':

                    s.push(op1 - op2);

                    break;

                case '\*':

                    s.push(op1 \* op2);

                    break;

                case '/':

                    s.push(op1 / op2);

                    break;

                default:

                    break;

            }

        }

    }

    return s.top();

}

int main() {

    string exp;

    cout << "Enter expression: ";

    getline(cin, exp);

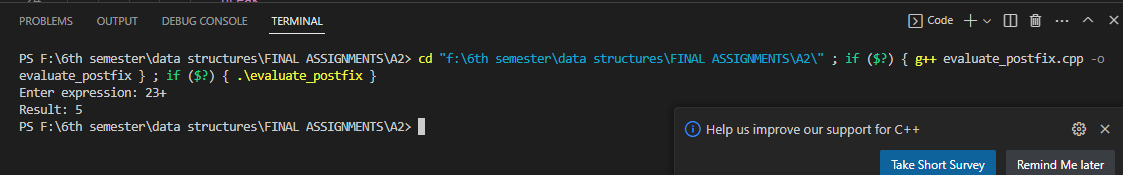
    int result = expression(exp);

    cout << "Result: " << result << endl;

    return 0;

}

Output:



Q2: Write a function that matches the brackets in a given string using a stack. If all the brackets

match then the function should return true, it should return false otherwise.

Code:

#include <iostream>

#include <stack>

#include <string>

using namespace std;

bool parenthesis(string s) {

    stack<char> string;

    for(char c : s)

    {

        if(c == '(' || c == '{' || c == '[')

        {

            string.push(c);

        }

        else if(c == ')' || c == '}' || c == ']')

        {

            if(string.empty())

            {

                return false;

            }

            else if((c == ')' && string.top() == '(') || (c == '}' && string.top() == '{') || (c == ']' && string.top() == '['))

            {

                string.pop();

            }

            else

            {

                return false;

            }

        }

    }

    return string.empty();

}

int main() {

    string s;

    cout << "Enter string: ";

    getline(cin, s);

    if(parenthesis(s)) {

        cout << "Parenthesis matched ." << endl;

    } else {

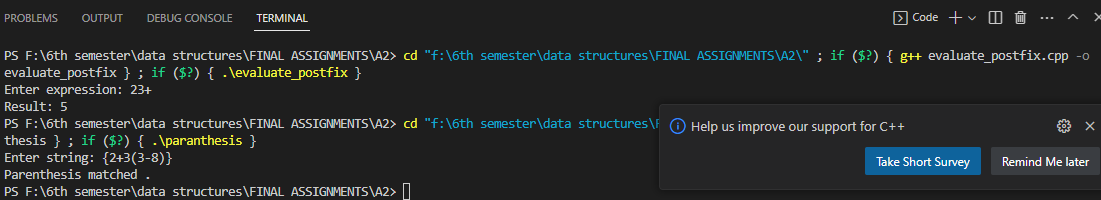
        cout << "Parenthesis left in stack." << endl;

    }

    return 0;

}

Output:

\_

The End\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_