```cpp
#include<bits/stdc++.h>

using namespace std;

class Quadruple {
public:
    std::string op;
    std::string arg1;
    std::string arg2;
    std::string result;

    Quadruple(std::string op, std::string arg1, std::string arg2, std::string
result)
        : op(op), arg1(arg1), arg2(arg2), result(result) {}
};

class QuadrupleGenerator {
private:
    std::vector<Quadruple> quadruples;
    int tempCounter;

    std::string getNextTemp() {
        return "t" + std::to_string(++tempCounter);
    }

public:
    QuadrupleGenerator() : tempCounter(0) {}

    void generateQuadruples(const std::string& postfix) {
        std::stack<std::string> operandStack;

        for (char c : postfix) {
            if (std::isalnum(c)) {
```

```
            operandStack.push(std::string(1, c));
        } else {
            std::string arg2 = operandStack.top(); operandStack.pop();
            std::string arg1 = operandStack.top(); operandStack.pop();
            std::string result = getNextTemp();
            quadruples.emplace_back(std::string(1, c), arg1, arg2,
result);
            operandStack.push(result);
        }
    }
}

void printQuadruples() {
    std::cout << "\nQuadruple Table:\n";
    std::cout << "-----------------------------\n";
    std::cout << "Op\tArg1\tArg2\tResult\n";
    std::cout << "-----------------------------\n";
    for (const auto& q : quadruples) {
        std::cout << q.op << "\t" << q.arg1 << "\t" << q.arg2 << "\t" <<
q.result << "\n";
    }
    std::cout << "-----------------------------\n";
}
};

class InfixToPostfixConverter
{
private:
    unordered_map<char, int> precedence;

    bool isOperator(char c)
    {
        return precedence.find(c) != precedence.end();
```

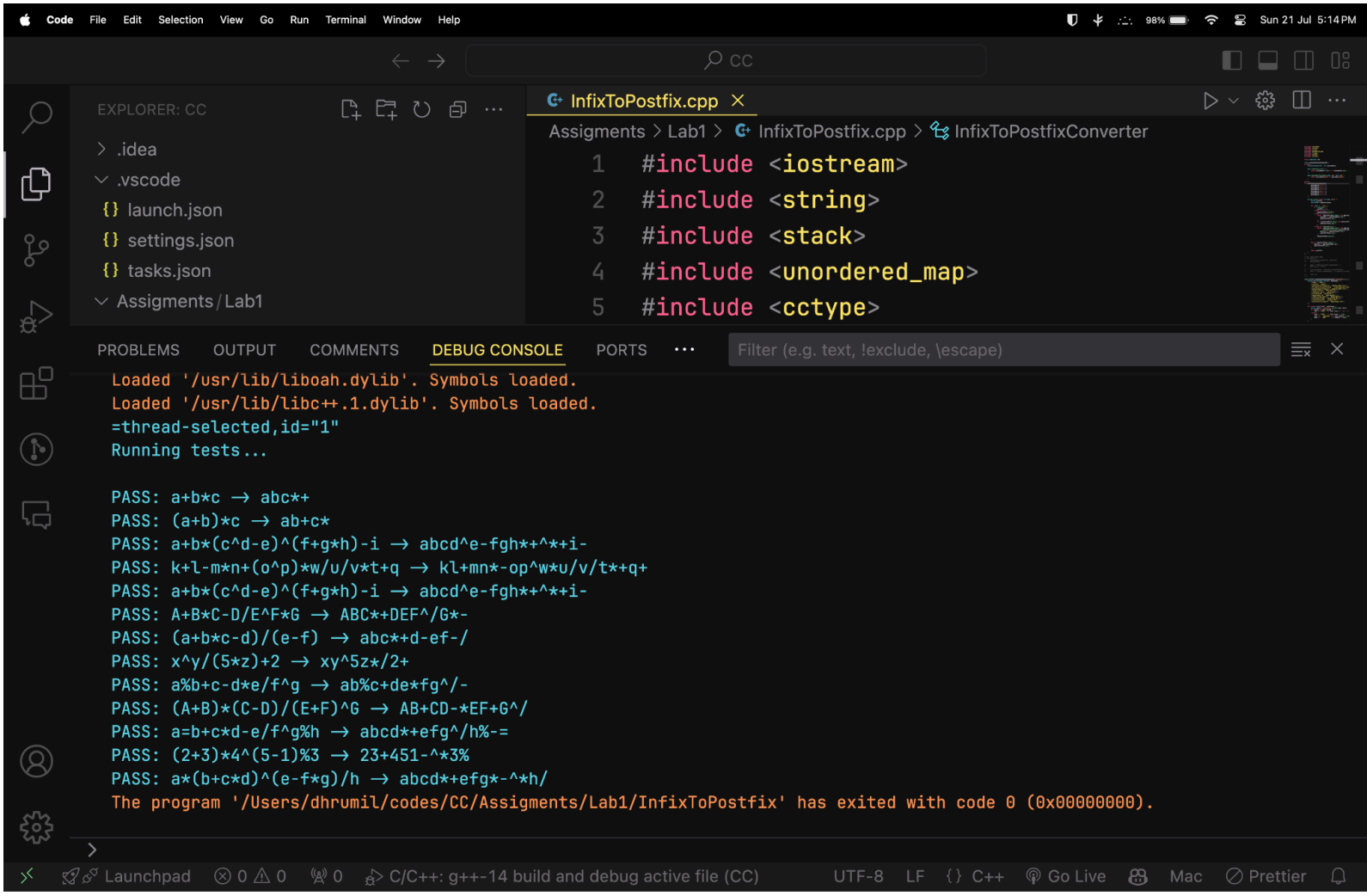
```
    }

    bool hasHigherPrecedence(char op1, char op2)
    {
        return precedence[op1] >= precedence[op2];
    }

public:
    InfixToPostfixConverter()
    {
        precedence['+'] = 1;
        precedence['-'] = 1;
        precedence['*'] = 2;
        precedence['/'] = 2;
        precedence['%'] = 2;
        precedence['^'] = 3;
        precedence['='] = 0;
    }

    string convert(const string &infix)
    {
        string postfix;
        stack<char> operatorStack;

        for (char c : infix)
        {
            if (isalnum(c))
            {
                postfix += c;
            }
            else if (c == '(')
            {
                operatorStack.push(c);
```

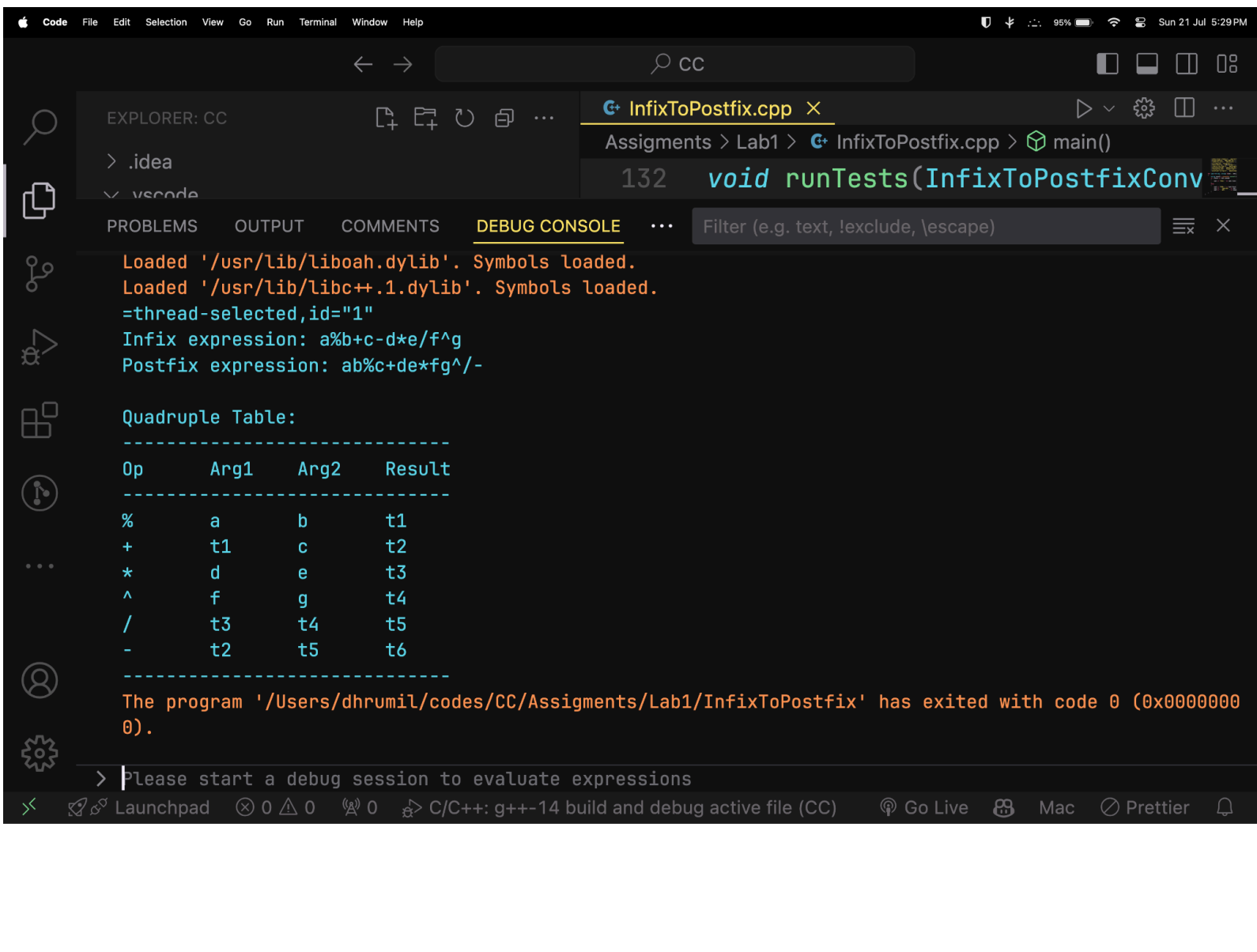
```
        }
        else if (c == ')')
        {
            while (!operatorStack.empty() && operatorStack.top() != '(')
            {
                postfix += operatorStack.top();
                operatorStack.pop();
            }
            if (!operatorStack.empty() && operatorStack.top() == '(')
            {
                operatorStack.pop();
            }
        }
        else if (isOperator(c))
        {
            while (!operatorStack.empty() && operatorStack.top() != '('
&&
                   hasHigherPrecedence(operatorStack.top(), c))
            {
                postfix += operatorStack.top();
                operatorStack.pop();
            }
            operatorStack.push(c);
        }
    }

    while (!operatorStack.empty())
    {
        postfix += operatorStack.top();
        operatorStack.pop();
    }

    return postfix;
```

```
    }
};

void runTests(InfixToPostfixConverter &converter)
{
    vector<pair<string, string>> testCases = {
        {"a+b*c", "abc*+"},
        {"(a+b)*c", "ab+c*"},
        {"a+b*(c^d-e)^(f+g*h)-i", "abcd^e-fgh*+^*+i-"},
        {"k+l-m*n+(o^p)*w/u/v*t+q", "kl+mn*-op^w*u/v/t*+q+"},
        {"a+b*(c^d-e)^(f+g*h)-i", "abcd^e-fgh*+^*+i-"},
        {"A+B*C-D/E^F*G", "ABC*+DEF^/G*-"},
        {"(a+b*c-d)/(e-f)", "abc*+d-ef-/"},
        {"x^y/(5*z)+2", "xy^5z*/2+"},
        {"a%b+c-d*e/f^g", "ab%c+de*fg^/-"},
        {"(A+B)*(C-D)/(E+F)^G", "AB+CD-*EF+G^/"},
        {"a=b+c*d-e/f^g%h", "abcd*+efg^/h%-="},
        {"(2+3)*4^(5-1)%3", "23+451-^*3%"},
        {"a*(b+c*d)^(e-f*g)/h", "abcd*+efg*-^*h/"}};

    for (pair<string, string> &test : testCases)
    {
        string result = converter.convert(test.first);
        if (result == test.second)
        {
            cout << "PASS: " << test.first << " -> " << result << endl;
        }
        else
        {
            cout << "FAIL: " << test.first << endl;
            cout << "  Expected: " << test.second << endl;
            cout << "  Got:      " << result << endl;
        }
```

```
    }
}

int main()
{
    InfixToPostfixConverter converter;
    QuadrupleGenerator generator;

    // cout << "Running tests...\n"
    //      << endl;
    // runTests(converter);


    string infix = "a%b+c-d*e/f^g";

    string postfix = converter.convert(infix);
    cout << "Infix expression: " << infix << endl;
    cout << "Postfix expression: " << postfix << endl;

    generator.generateQuadruples(postfix);
    generator.printQuadruples();
    return 0;
}

// for interactive mode
// int main() {
//     InfixToPostfixConverter converter;
//     string infix;

//     cout << "Enter an infix expression: ";
//     getline(cin, infix);

//     string postfix = converter.convert(infix);
```

```
//     cout << "Postfix expression: " << postfix << endl;

//     return 0;
// }
```


Code File Edit Selection View Go Run Terminal Window Help
Sun 21 Jul 5:14 PM
cc
EXPLORER: CC
.idea
.vscode
launch.json
settings.json
tasks.json
Assigments / Lab1
InfixToPostfix.cpp
Assigments > Lab1 > InfixToPostfix.cpp > InfixToPostfixConverter
1 #include <iostream>
2 #include <string>
3 #include <stack>
4 #include <unordered_map>
5 #include <cctype>
PROBLEMS OUTPUT COMMENTS DEBUG CONSOLE PORTS
Filter (e.g. text, !exclude, \escape)
Loaded '/usr/lib/liboah.dylib'. Symbols loaded.
Loaded '/usr/lib/libc++.1.dylib'. Symbols loaded.
=thread-selected,id="1"
Running tests...
PASS: a+b*c → abc*+
PASS: (a+b)*c → ab+c*
PASS: a+b*(c^d-e)^(f+g*h)-i → abcd^e-fgh*+^*+i-
PASS: k+l-m*n+(o^p)*w/u/v*t+q → kl+mn*-op^w*u/v/t*+q+
PASS: a+b*(c^d-e)^(f+g*h)-i → abcd^e-fgh*+^*+i-
PASS: A+B*C-D/E^F*G → ABC*+DEF^/G*-
PASS: (a+b*c-d)/(e-f) → abc*+d-ef-/
PASS: x^y/(5*z)+2 → xy^5z*/2+
PASS: a%b+c-d*e/f^g → ab%c+de*fg^/-
PASS: (A+B)*(C-D)/(E+F)^G → AB+CD-*EF+G^/
PASS: a=b+c*d-e/f^g%h → abcd*+efg^/h%-=
PASS: (2+3)*4^(5-1)%3 → 23+451-^*3%
PASS: a*(b+c*d)^(e-f*g)/h → abcd*+efg*-^*h/
The program '/Users/dhrumil/codes/CC/Assigments/Lab1/InfixToPostfix' has exited with code 0 (0x00000000).
Launchpad 0 0 0 C/C++: g++-14 build and debug active file (CC)
UTF-8 LF {} C++ Go Live Mac Prettier


Code File Edit Selection View Go Run Terminal Window Help
Sun 21 Jul 5:29 PM
cc
EXPLORER: CC
.idea
InfixToPostfix.cpp
Assigments > Lab1 > InfixToPostfix.cpp > main()
132 void runTests(InfixToPostfixConv
PROBLEMS OUTPUT COMMENTS DEBUG CONSOLE
Filter (e.g. text, !exclude, \escape)
Loaded '/usr/lib/liboah.dylib'. Symbols loaded.
Loaded '/usr/lib/libc++.1.dylib'. Symbols loaded.
=thread-selected,id="1"
Infix expression: a%b+c-d*e/f^g
Postfix expression: ab%c+de*fg^/-
Quadruple Table:
Op Arg1 Arg2 Result
% a b t1
+ t1 c t2
* d e t3
^ f g t4
/ t3 t4 t5
- t2 t5 t6
The program '/Users/dhrumil/codes/CC/Assigments/Lab1/InfixToPostfix' has exited with code 0 (0x00000000).
Please start a debug session to evaluate expressions
Launchpad 0 0 0 C/C++: g++-14 build and debug active file (CC)
Go Live Mac Prettier