Input:

```
#include <iostream>
#include <string>
#include <vector>
#include <regex>
#include <stack>
using namespace std;
void analyzeExpression(const string& input);
bool isValidExpression(const string& expression);
bool areParenthesesBalanced(const string& expression);
int evaluateExpression(const vector<string>& tokens);
int main() {
    string input;
    cout << "Enter an arithmetic expression:" << endl;
    getline(cin, input);
    analyzeExpression(input);
    return 0;
void analyzeExpression(const string& input) {
    cout << "\nLexical Analysis:" << endl;
    vector<string> tokens;
    string token;
    for (char c : input) {
        if (c == '+' || c == '-' || c == '*' || c == '/' || c == '(' || c == ')') {
            if (!token.empty()) {
                tokens.push_back(token);
                token.clear();
            }
            tokens.push_back(string(1, c));
        } else if (c >= '0' && c <= '9') {
            token += c;
        }
    }
```

```
if (!token.empty()) {
    tokens.push_back(token);
}
for (const string& t : tokens) {
    cout << "Token: " << t << endl;
}
cout << "\nNumber of Tokens: " << tokens.size() << endl;
cout << "\nSyntax Analysis:" << endl;
bool syntaxCorrect = true;
int openParentheses = 0;
for (const string& t : tokens) {
    if (t == "(") {
        openParentheses++;
    } else if (t == ")") {
        openParentheses--;
        if (openParentheses < 0) {
            syntaxCorrect = false;
            break;
        }
    }
}
if (openParentheses != 0) {
    syntaxCorrect = false;
}
if (syntaxCorrect) {
    cout << "Syntax is correct." << endl;
} else {
    cout << "Syntax error: unbalanced parentheses." << endl;
    return;
}
cout << "\nSemantic Analysis:" << endl;
if (!isValidExpression(input)) {
```

```
        cout << "Error: Invalid expression." << endl;
        return;
    }
    cout << "\nCode Generation:" << endl;
    for (const string& t : tokens) {
        cout << t << " ";
    }
    cout << endl;
    int result = evaluateExpression(tokens);
    cout << "\nResult: " << result << endl;
}
bool isValidExpression(const string& expression) {
    string temp = regex_replace(expression, regex("\\s+"), "");
    if (!regex_match(temp, regex("[0-9+\\-*/()]+"))) {
        return false;
    }
    if (!areParenthesesBalanced(expression)) {
        return false;
    }
    return true;
}
bool areParenthesesBalanced(const string& expression) {
    int count = 0;
    for (char c : expression) {
        if (c == '(') {
            count++;
        } else if (c == ')') {
            count--;
            if (count < 0) {
                return false;
            }
        }
```

```
    }
    return count == 0;
}
int evaluateExpression(const vector<string>& tokens) {
    stack<int> operands;
    stack<char> operators;
    for (const string& token : tokens) {
        if (token == "+" || token == "-" || token == "*" || token == "/") {
            operators.push(token[0]);
        } else if (token == "(") {
            operators.push('(');
        } else if (token == ")") {
            while (!operators.empty() && operators.top() != '(') {
                char op = operators.top();
                operators.pop();
                int operand2 = operands.top();
                operands.pop();
                int operand1 = operands.top();
                operands.pop();
                if (op == '+') {
                    operands.push(operand1 + operand2);
                } else if (op == '-') {
                    operands.push(operand1 - operand2);
                } else if (op == '*') {
                    operands.push(operand1 * operand2);
                } else if (op == '/') {
                    operands.push(operand1 / operand2);
                }
            }
            operators.pop(); // Pop '('
        } else {
            operands.push(stoi(token));
```

```
        }
    }
    while (!operators.empty()) {
        char op = operators.top();
        operators.pop();
        int operand2 = operands.top();
        operands.pop();
        int operand1 = operands.top();
        operands.pop();
        if (op == '+') {
            operands.push(operand1 + operand2);
        } else if (op == '-') {
            operands.push(operand1 - operand2);
        } else if (op == '*') {
            operands.push(operand1 * operand2);
        } else if (op == '/') {
            operands.push(operand1 / operand2);
        }
    }
    return operands.top();
}
```

Output: