CC - 213 : DATA STRUCTURES AND ALGORITHMS

BSCS MORNING - FALL 2022

ASSIGNMENT # 01

TOPIC:

INFIX TO POSTFIX CONVERSION AND EVALUATION

Objective:

Implement functions to convert infix expressions to postfix and evaluate these postfix expressions using stacks and a linked list to store variable values.

Problem Statement:

You are required to write a program in C++ that can:

- Convert an infix expression to a postfix expression.
- Evaluate a postfix expression.

Your program should handle variable names and constants. Assume the variable values are provided by the user during execution.

Requirements:

Class Definition:

Define a class **VariableNode** with the following data members:

- string variableName
- int value

Implement a linked list to store instances of **VariableNode**.

Infix to Postfix Conversion:

Implement a function string **infixToPostfix**(string expression) that converts an infix expression to a postfix expression.

The expression can contain the operators +, -, *, /, and parentheses (and).

Variables are alphanumeric strings (e.g., a, b1, var2). Constants are integers.

Postfix Evaluation:

Implement a function int **evaluatePostfix**(string postfixExpression, **VariableNode*** head) that evaluates a postfix expression given the head of the linked list of variable values.

Prompt the user to input values for the variables in the expression.

Variable Extraction:

Implement a function to extract variables from infix expression and ask user for their values and store variables along their values in a list of **VariableNode**.

Sample Code:

```
#include <iostream>
#include <string>
using namespace std;
class VariableNode
{
public:
    string variableName;
    int value;
    VariableNode *next;
    VariableNode(string name, int val) : variableName(name), value(val), next(nullptr) {}
};
string infixToPostfix(string expression)
{
    // Your implementation here
    // Hint: Use a stack to handle operators and parentheses
}
int evaluatePostfix(string postfixExpression, VariableNode *variableList)
{
    // Your implementation here
    // Hint: Use a stack to evaluate the postfix expression
}
VariableNode * extractVariables(string expression)
{
    // your implementation here
    // Hint: Use variableNode to implement a list of variables along with their
values(input from user) while extracting them.
}

int main()
{
    string infixExpression = "a + b * (c - d)";
    string postfixExpression = infixToPostfix(infixExpression);
    VariableNode* variables = extractVariables(infixExpression);
    int result = evaluatePostfix(postfixExpression, variables);
    cout << "Evaluation Result: " << result << endl;
    return 0;
}
```

Sample Output 1:

```
Infix Expression: (temperatureFahrenheit - 32) * 5/9

Enter value for temperatureFahreheit: 50

Evaluation Result: 10
```

Sample Output 2:

```
Infix Expression: (m1 + 1) * n2 - 3 / r

Enter value for m1: 2

Enter value for n2: 4

Enter value for r: 2

Evaluation Result: 11
```

Sample Output 3:

```
Infix Expression: ((obtainedMarks + bonus) / totalMarks) * 100

Enter value for obtainedMarks: 50

Enter value for bonus: 10

Enter value for totalMarks: 100

Evaluation Result: 60
```

Guidelines:

- Use your own implemented stack class to help with the conversion from infix to postfix.
- Ensure proper handling of operator precedence and associativity.
- For evaluation, use your own implemented stack class to evaluate the postfix expression.
- Implement a linked list to store variable values.
- Prompt the user to enter values for each variable found in the expression.

Submission:

- Submit your solution as a zip file containing only .h and .cpp files named BCSF22M0XX-DSA-ASSIGNMENT-1.zip
- Include comments explaining your code and logic.
- Ensure your code is readable and follows good programming practices.

Happy coding!