CS 280
Spring 2023
Recitation Assignment 7

March 30, 2023

Due Date: Monday, April 3rd, 2023, 23:59
Total Points: 8

A postfix expression is defined as an arithmetic expression where the operator occurs after the operands. For example, using infix notation we would add **4** and **5** using **4 + 5**. In postfix notation we would arrange the operator to occur at the end: **4 5 +**. A postfix expression can become arbitrarily complex such as **3 4 5 + − 2 ***. This expression would be expressed using infix notation like this: **((4 + 5) − 3) * 2**. As you can see, the postfix form does not require parentheses because the order of operations is made explicit. It turns out that evaluating postfix expressions is quite easy when combined with a stack data structure. We read a postfix expression from left to right pushing values onto the stack when a number is encountered and popping numbers from the stack when an operator is encountered. For example, the expression **4 5 +** would require the following operations:

1. Push 4 onto the stack.
2. Push 5 onto the stack.
3. Pop the top two values of the stack, add them, and then push the result back onto the stack.

Write a C++ function that allows a user to evaluate postfix expressions. In this assignment, postfix expressions are described by using integer constants and variables (restricted to one letter identifiers), as operands, and supports the basic arithmetic operators such as +, −, *, and /, for addition, subtraction, multiplication, and division, respectively. To allow using variables, you should use the **map** container from the STL to map variable names (e.g., X) to integer values. The postfix expression evaluator supports variables as follows. When a variable is encountered from the input, it should pop the current value off the top of the stack and associate it with that variable in the map. If the variable is preceded by a '$', the postfix evaluator should simply retrieve the variable's associated value from the map container and push it onto the stack. The function scans the input string left to right. When scanning input is completed, the function pops the top of the stack for the result, if it is not empty. If the stack is empty, the function displays the following message, then returns.

```
Error: Incomplete input postfix expression.
```

The popped value from the stack is the final result, and can be printed out if the stack becomes empty afterwards. If the stack does not become empty after popping the final result, then there is an error and the function should display the following message and return back.

```
The evaluation is incomplete, missing input operators.
```

For example, the evaluation of the following simple postfix expression produces the result 27 to be displayed on the console.

```
4 5 + X $X 3 *
```

Given the above expression, the function should print out the result as follows:

```
The result of evaluating the postfix expression "4 5 + X $X 3 *"
is the value: 27
```

For an incomplete/invalid postfix expression similar to the following expression:
`X 4 5 +`

the function should display the error message:
`Error: Incomplete input postfix expression.`

Implement the C++ postfix expression evaluator based on the following header definition:

```
void PostfixEval(string instr);
```

Where, the `instr` is the passed input string to the function. The postfix expression evaluator should check for errors due to illegal operator symbol, incomplete input postfix expression, or invalid string/identifier. and print out messages as described in the examples shown above and in the slides.

Vocareum Automatic Grading

- A driver program is provided for testing the implementation, called "ra7prog.cpp", on Vocareum. The "ra7prog.cpp" will be propagated to your Work directory. The program reads from the keyboard an input string, and makes a call to the `PostfixEval()`.
- You are provided by a set of 7 testing files associated with Recitation Assignment 7. Vocareum automatic grading will be based on these testing files. You may use them to check and test your implementation. These are available in a compressed archive "RA7 Test Cases.zip" on Canvas assignment. The testing case of each file is defined in the Grading table below.
- "ra7prog.cpp" is available with the other assignment material on Canvas.

Submission Guidelines

- Please upload your implementation to Vocareum as a "`PostfixEval`.cpp" file. The file should include the implementation of the function `PostfixEval()`.

- **Submissions after the due date are accepted with a fixed penalty of 25% from the student's score. No submission is accepted after Wednesday 11:59 pm, April 5, 2023.**

Grading Table

Item	Points
Compiles Successfully	1
String 1	1
String 2	1
String 3	1
String 4	1
String 5	1
String 6	1
String 7	1
Total	8