San José State University
Department of Computer Engineering

CMPE 180A
Data Structures and Algorithms in C++

Spring 2018

Instructor: Ron Mak

Assignment #10

Assigned: Thursday, April 5
Due: Thursday, April 12 at 5:30 PM
Canvas: Assignment #10. Calculator
Points: 100 with extra credit

Calculator

This assignment will give you practice with mutual recursion. Write a **`Calculator`** class that prompts for and evaluates arithmetic expressions and then prints their results. An expression contains numbers and the operators **`+ - * /`** and **`/`**. An expression can also contain parenthesized subexpressions nested arbitrarily deeply.

Syntax diagrams

Use these syntax diagrams to help you design your **`Calculator`** member functions.

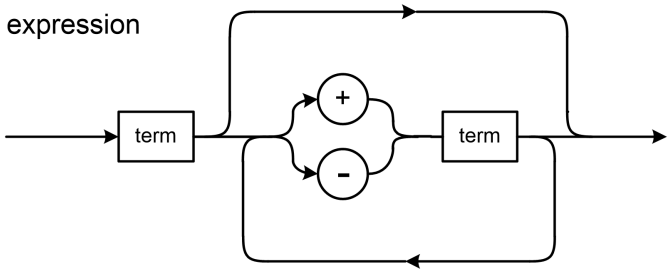
An expression is either a single term, or a term followed by more terms separated by **+** or **–** operators.

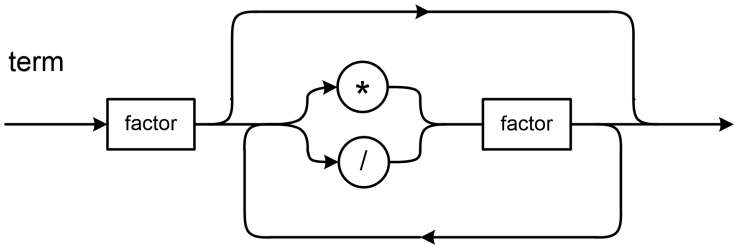
A term is either a single factor, or a factor followed by more factors separated by ***** or **/** operators.

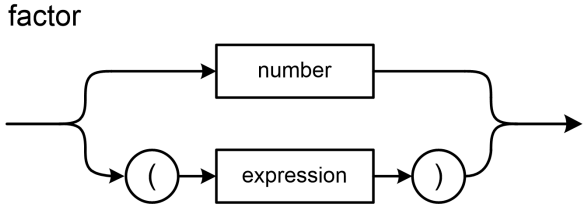
A factor is either a number or a parenthesized expression. Expressions can be nested arbitrarily deep.

A number is a C++ double that can be read by **>>**

Require each expression end with an **=** sign. Ignore blanks in the expressions.







Precedence rules

Your program must abide by the normal precedence rules: Operators `*` and `/` have higher precedence than `+` and `-`. Evaluate parenthesized subexpressions first, and when there are nested parentheses, evaluate the innermost subexpressions first. *If your program follows the syntax diagrams, it will implement the precedence rules correctly.*

If there are several consecutive operators at the same precedence level and there are no parentheses, evaluate from left to right. Therefore, for `1 + 2 + 3 + 4` add 3 to the sum of 1 and 2 and then add 4 to that sum.

Error handling

Your calculator should catch syntax errors such as invalid operators or unbalanced parentheses. It should catch attempts to divide by zero. Print an appropriate error message, ignore the rest of the expression, and prompt for the next expression.

Sample output

Here is a sample interaction with the calculator program. You choose what expressions to enter, but they should show off the capabilities of your program. Include expressions that contain errors.

```
Expression? 42 =
42

Expression? 2*(42 - 42e-1) =
75.6

Expression? 12/(5.5*(17 + 0.00314E+3)) =
0.108333

Expression? 65%2 =
65
***** Unexpected %

Expression? 12/(5.5*(17 + 0.00314E+3) =
***** Missing )

Expression? (((((5))))) =
5

Expression? (5)) + 2 =
5
***** Unexpected )

Expression? 5/(6 - 2*3) =
***** Division by zero

Expression? .

Done!
```

Program design

Create a **`Calculator`** class with appropriate member functions expression, term, and factor. Write the **`main`** in a separate file.

No CodeCheck

This program is interactive, and you choose the expressions to give it and the format of its prompts and answers. Therefore, you will not use CodeCheck for this assignment.

This also means that *you will write the program in its entirety*.

Submission into Canvas

Cut and paste into a text file a sample run of your program with a good sample of expressions. Zip this text file along all your C++ source files. Name the zip file after yourself and submit it into Canvas: **Assignment #10. Calculator.**

Rubric

Your program will be graded according to these criteria:

Criteria	Max points
Good program execution • User interaction (prompts for expressions, etc.) • Correct evaluation of expressions • Good examples of program capabilities (all operators, nested parenthesized subexpressions, etc.) • Error handling (invalid operator, division by zero, etc.)	**40** • 10 • 10 • 10 • 10
Good program design • Class **`Calculator`** • Member functions **`expression`**, **`term`**, and **`factor`** • Separate calculator test program (with **`main`**)	**30** • 10 • 10 • 10
Good program style • Descriptive variable names. • Meaningful comments. • Follow the coding style (formatting, braces, indentation, function declarations before the main, etc.) of the Savitch textbook.	**30** • 10 • 10 • 10

Extra credit #1 (up to 10 points)

Allow a leading - (minus) sign to negate a value. Examples: **`-5  -2*3  -(4+5)`**
For completeness, you should also allow a leading **+** sign which will have no effect. You should not allow an operator next to a leading **+** or **−** sign (blanks don't count as separators). Examples of invalid expressions: **`2*-3  6--2  7/+3`**

Tip: You only need to modify one of the syntax diagrams to allow the optional leading **+** or **−** sign.

Extra credit #2 (up to 25 points)

Implement exponentiation with the **^** operator. For example, **`2^3`** is 2^3. Add member function **`power`**.

Exponentiation has the highest precedence. For example, evaluate **`1+2*3^4`** as **`1+(2*(3^4))`**

Consecutive exponentiation in the absence of parentheses is evaluated right to left. Therefore evaluate **`2^3^4`** as **`2^(3^4)`**, i.e., 2^{3^4}

Modify the syntax diagrams to accommodate the exponentiation operator. Your new diagrams should give the **^** operator the highest precedence and show that it is evaluated right to left.

Example output with both extra credits:

```
Expression? -5=
-5

Expression? +5=
5

Expression? -2*3=
-6

Expression? 4*(-5+2)=
-12

Expression? -(2+3)=
-5

Expression? 5+(-3)=
2

Expression? 6--3=
***** Invalid factor

Expression? 3*-2=
***** Invalid factor

Expression? 3/ +2=
***** Invalid factor
```

```
Expression? 2^3=
8

Expression? 2^3^4=
2.41785e+24

Expression? (2^3)^4=
4096

Expression? 64^0.5=
8

Expression? 256^(1/4)=
4

Expression? 256^(-1/4)=
0.25

Expression? .

Done!
```

Tip: Look up the C++ **`pow`** function.

Academic integrity

You may study together and discuss the assignments, but what you turn in must be your individual work. Assignment submissions will be checked for plagiarism using Moss (http://theory.stanford.edu/~aiken/moss/). **Copying another student's program or sharing your program is a violation of academic integrity.** Moss is not fooled by renaming variables, reformatting source code, or re-ordering functions.

Violators of academic integrity will suffer severe sanctions, including academic probation. Students who are on academic probation are not eligible for work as instructional assistants in the university or for internships at local companies.