

Advanced Programming

Programming Assignment #2



Kang Hoon Lee

Kwangwoon University

Basic Requirements

- ❑ **Fix every syntactic & logical error in `calculator_buggy.cpp`**
 - Basic functionalities + unary minus + variables
 - E.g., use cases #1

 - ❑ **Extend your code to handle `unary plus` as well**
 - E.g., use cases #2

 - ❑ **Pre-declare the following two variables in your code so that those can be used from the beginning without declaration**
 - `“pi” = 3.14159, “e” = 2.71828`
 - E.g., use cases #3
-

Use Cases #1

> 3+5;

= 8

> 2*(5-3);

= 4

> -2*-2;

= 4

> +2*+2;

primary expected

> let width=3;

= 3

> let height=4;

= 4

> width*height;

= 12

Use Cases #2

> +2;

= 2

> +2+2;

= 4

> 2*+2;

= 4

> -2++2;

= 0

> +2--2;

= 4

> let a=+1;

= 1

> let b=-1;

= -1

Use Cases #3

```
> pi;  
= 3.14159  
> let r=5;  
= 5  
> 2*pi*r;  
= 31.4159  
> let area=pi*r*r;  
= 78.5397  
> let e=2.7;  
e declared twice  
> e;  
= 2.71828  
> pi*e;  
= 8.53972
```

Advanced Requirements

- **Add an exponentiation operator**
 - Use a binary \wedge Operator to represent “exponentiation”
 - When the right operand is a positive integer, an exponentiation can be rewritten as a repeated multiplication
 - E.g., the expression 2^3 means $2*2*2$
 - In general, when the right operand is an arbitrary real number, you can evaluate the result by using C++ standard `pow()` function
 - Make \wedge operator bind tighter than $*$ and $/$
 - E.g., $2*2^3$ means $2*(2^3)$ rather than $(2*2)^3$

Hint: Begin by modifying the grammar to account for a higher-level operator
 - Make \wedge operator right-associative
 - E.g., 2^{2^3} means $2^{(2^3)}$ rather than $(2^2)^3$

Hint: For any successive operations of exponentiation (e.g., a^{b^c}), use **vector** to store operands left-to-right ($[a, b, c]$), and later combine those in the reverse order ($b' = \text{pow}(b, c)$, $a' = \text{pow}(a, b')$)

Advanced Requirements

☐ Add mathematical functions

- Allow the user to use a set of mathematical functions including `sqrt()`, `sin()`, `cos()`, and `tan()`
 - ☐ E.g., `sqrt(9)` is `3`, `sin(0)` is `0`, `cos(pi)` is `-1`, and so on
- Use the standard library math functions that are available through the header `std_lib_facilities.h`
- Catch attempts to give invalid arguments, such as negative number for `sqrt()`, and print appropriate error messages

☐ (Optional) Add any other useful features for your calculator

- Describe what features are additionally supported, and how those are implemented in detail in your report
-

Note

☐ Code (*.cpp)

- The common header file **std_lib_facilities.h** and your *.cpp file should be in the same folder (project folder)
- Include only std_lib_facilities.h and **do not include any other *.h**

☐ Report (*.pdf)

- Title page
 - ☐ Course title, submission date, affiliation, student ID, full name
 - Begin with a summary of your results
 - ☐ Which requirements did you fulfill? And which didn't you? (present a simple table)
 - ☐ Did you implement some additional features? What are those?
 - For each requirement (basic/advanced/optional), explain how you fulfilled it
 - ☐ Do not just dump the entire code
 - ☐ It's okay to copy snippets of your code to complement written description
 - Conclude with some comments on your work
 - ☐ Key challenges you have successfully tackled
 - ☐ Limitations you hope to address in the future
-

Submission

- ☐ **Compress your code and report into a single *.zip file**
 - **Code**
 - ☐ The entire project folder including *.sln, *.cpp, *.h, etc.
 - ❖ The grader should be able to open the *.sln with Visual Studio 2019 and build/run the project immediately without any problems
 - **Report**
 - ☐ A single *.pdf file
 - ❖ You should convert your word format (*.hwp, *.doc, *.docx) to PDF format (*.pdf) before zipping
 - **Name your zip file as your student ID**
 - ❖ ex) 2012726055.zip
 - ☐ **Upload to homework assignment in U-campus**
 - ☐ **Due at 5/17 (Sun), 11:59 PM**
-