Programming Assignment 2

Computer Programming for Engineers (DASF003-41)

Instructor: Sungjae Hwang

TAs: Bohyun Lee, Kyongshik Lee

Introduction

■Deadline : 2022.11.20

■ You have two days for late submission (~2022.11.22)

- **25%** deduction per day

■Submit both source code and Makefile

- You will not get a point if your makefile do not build an executable program

Problem 1 (50pt)

■Description

1. **Your task is to develop a class named “Fraction” which represents fraction values.**
2. **Fraction class contains three private variables: N (Integer), D (Denominator), NU (Numerator).**
3. $N, NU \in \{\mathbb{N}, 0\}, D \in \mathbb{N}$

```
int N;          // Integer
int D;          // Denominator
int NU;         // Numerator
```

→ $N\frac{NU}{D}$

■Functions

Fraction class contains following member functions.

Functions Signature	Explanation
Fraction sum (Fraction b)	Add two fractions and return result
Fraction sum (double b)	Add double and fraction value and return result
Fraction multiply (Fraction b)	Multiply two fractions and return result
Fraction multiply (double b)	Multiply double and fraction value and return result

Problem 1 (50pt)

■Functions

Fraction class contains following member functions.

Functions Signature	Explanation
void abbreviation ()	Abbreviate fraction.
Bool toMixedNum ()	Convert fraction into mixed number. - Updates N, D, and NU. - If it is changed, return true, else return false.
void print ()	Print fraction value * Format : N and NU/D
double toDouble ()	Convert fraction into double. * Round to the nearest sixth decimal place (e.g., 0.0000167->0.000017)
Fraction str2Fraction (string str)	Convert string into a Fraction class * String format : N/NU/D
Fraction double2Fraction (double val)	Convert double value into a Fraction class

Problem 1 (50pt)

■Input

frac1 frac2

*** frac1: Fraction expressed in string (e.x "1/1/2" where the format is N/NU/D)**

*** frac2: double value (e.x. 1.25)**

■Output

The program produces 10 lines of output based on the given input

```
frac1
frac2 (=double2Fraction(d2))
frac1.sum(frac1) (++)
frac1.sum(d2) (++)
frac2.multiply(frac1) (++)
frac2.multiply(d2) (++)
frac1.toDouble()
frac2.toDouble()
```

Output Format

Input	Output
1/1/2 1.25	1 and 1/2 1 and 1/4 3 and 0/0 2 and 3/4 1 and 7/8 1 and 9/16 1.500000 1.250000

Input	Output
1/3/5 1.00625	1 and 3/5 1 and 1/160 3 and 1/5 2 and 97/160 1 and 61/100 1 and 321/25600 1.600000 1.006250

Concrete Output Example

Problem 1 (50pt)

■Restriction

1. All returned fractions should be abbreviated form
2. All improper fraction should be converted into mixed number form
3. If there is no integer part in fraction, set N=0
4. If denominator is 1, set D and NU to 0 and updates N.
5. You cannot use another library except iostream and string
6. Print “Incorrect Input” if the input is the wrong format or if it is not numbers.
7. If the denominator is 0, treat it as incorrect input. (++)
8. Double input is greater than 0 and can be entered up to the sixth digit below the decimal point. (++)
9. If you don't write a function according to Functions Signature, it can be a big deduction factor for scoring. (++)
10. toMixedNum : Addition and multiplication operations may produce results in the form of improper fraction. Alternatively, results in a form other than a irreducible fraction may come out. In preparation for this situation, we asked you to define this function to update the result values. This function was removed from Output Format. (++)
11. Input validation can proceed in the function str2Fraction (string str), double2Fraction (double val). If you don't do input validation, -10pts. (++)

Problem 1 (50pt)

■Submission Files

- **Makefile**
- **main.cc**
- **fraction.h**
- **fraction.cc**

fraction.h file contains the class definition.

*** The class definition only includes member variables and declaration of member functions**

fraction.cc contains the implementation of the member functions.

Problem 1 (50pt)

■ Evaluation

- **5 points for each member functions listed in “Function” section.**
  - *** Total 10*5 = 50 points**
  - *** You must implement all 10 member functions.**
  - *** We give you the points based on these member functions.**
- **You can only use iostream and string library. 0 point if you use other libraries**
- **0 point if we cannot compile your program using the given makefile.**
  - **Your Makefile should make executable file in the same directory as your source code. (++)**
  - **Executable file name should be “fraction” (++)**
  - **Don’t modify folder name. (++)**

■ Math terms in Korean

- **Fraction :** 분수
- **Denominator :** 분모
- **Numerator :** 분자
- **Abbreviate :** 약분
- **Improper fraction :** 가분수
- **Mixed number :** 대분수

Problem 2 (50pt)

■Description

1. Your task is to develop a simple calculator to be able to calculate six operators.
2. The six operations : addition (+), subtraction (-), multiplication (*), division (/), remainder (%), and exponentiation (^).
3. The remainder is the integer “left over” after dividing one integer by another. (e.x. 13%3 would be 1, because the quotient of the equation is 4, and its remainder becomes 1 due to 13-(3*4).
4. Exponentiation is a mathematical operation, producing n times of multiplication of b, b^n, where a base (b) and a power (n) are given. (e.x. 3^4 = 3*3*3*3 = 81.)
5. The calculator supports an assignment (=) operation. Then, the calculator allows one to allocate up to three variables, x, y, and z for further use. (e.x. It is possible to enter x = 10, followed by x+20.
6. Each variable should be initialized as 0 in the beginning. In case of wrong variables given like “a” or “xy”, your program should emit “Invalid input”.

Problem 2 (50pt)

■Description

7. The calculator takes a single equation as an input at a time for evaluation.
   * e.x. if one wants to multiply two numbers, the input equation must be 3*4.
   * Likewise, 3%8 and 7^4 are examples of possible inputs.
   * The input may or may not have a space between a number and an operator.
8. The program keeps evaluating or computing with a single equation at a time. If a user enters quit, then the program terminates.
9. The input number must be between 0 to 255. When calculating the exponent operation, both the bottom and the index must be between 0 and 10. (if exception occurs, check example 6) (++)

Problem 2 (50pt)

■Restriction

1. The type of two input numbers are positive numbers with the type of either int or float for addition, subtraction, multiplication, and division.
2. The input numbers are treated as int only for remainder and exponentiation. For example, 9.5%4 would be regarded as 9%4, and 2.5^3.5 would be regarded as 2.5^3 (++).
3. The results of all operations can be expressed with int and float.
4. Division by zero must be avoided, that is, your code should raise an error with a message, “Operation disallowed”.
5. Valid form of input are :
   - Positive Number (+|-|*|/|%|^) Positive Number
   - x|y|z for assignment variables
   - “|” means “or”
6. We consider two operands and one operator only.
   - There are no test cases such as 1+1+1 and 1*2/3
   - So, you don’t need to validate them.
7. But, you need to validate the variable name. Only x,y,z are allowed.
   - Check example 8 of “Program Input & Output”

Problem 2 (50pt)

■Program Input & Output

- **Example 1**

```
./calculator
Enter your equation to calculate: 5 + 9
Answer: 14
Enter your equation to calculate: 7.5 - 20
Answer: -12.5
Enter your equation to calculate: 82 * 14
Answer: 1148
Enter your equation to calculate: quit
Sungjaeui-MBP:2021FALL_CPE sungjaehwang$
```

- **"calculator" is the program name**
- **"Enter your equation to calculate:" is the program output**
- ***"5 + 9", "7.5 - 20", "82 * 14", "quit"* are the inputs**
- **"5+9" without a space is allowed**
- **Program terminates if "quit" is entered**

Problem 2 (50pt)

■Program Input & Output

- **Example 2**

```
./calculator
Enter your equation to calculate: 144.0 / 12
Answer: 12.0
Enter your equation to calculate: 8 / 0
Answer: Operation disallowed
Sungjaeui-MBP:2021FALL_CPE sungjaehwang$
```

- **Program terminates with a wrong operation**

Problem 2 (50pt)

■Program Input & Output

- **Example 3**

```
./calculator
Enter your equation to calculate: 30.9 % 7
Answer: 2
Enter your equation to calculate: 5 ^ 3
Answer: 125
Enter your equation to calculate: 2.5 ^ 3
Answer: 15.625
Enter your equation to calculate: quit
Sungjaeui-MBP:2021FALL_CPE sungjaehwang$
```

- **The examples for remainder and exponentiation operations**

Problem 2 (50pt)

■Program Input & Output

- **Example 4**

```
./calculator
Enter your equation to calculate: x = 9.3
Enter your equation to calculate: x
Answer: 9.3
Enter your equation to calculate: quit
Sungjaeui-MBP:2021FALL_CPE sungjaehwang$
```

- **The examples for assignment operation**

Problem 2 (50pt)

■Program Input & Output

- **Example 5**

```
./calculator
Enter your equation to calculate: x = 3
Enter your equation to calculate: x + 4
Answer: 7
Enter your equation to calculate: x = 6
Enter your equation to calculate: y = 7
Enter your equation to calculate: x * y
Answer: 42
Enter your equation to calculate: z
Answer: 0
Enter your equation to calculate: quit
Sungjaeui-MBP:2021FALL_CPE sungjaehwang$
```

- **More examples for assignment operation**

Problem 2 (50pt)

■Program Input & Output

- Example 6

```
./calculator
Enter your equation to calculate: a
Answer: Invalid input
Sungjaeui-MBP:2021FALL_CPE sungjaehwang$
```

- Example of wrong variable
- Program should terminates with “Invalid input” message

■Submission Files

- Makefile
- main.cc
- calculator.h
- calculator.cc

calculator.h file contains the class definition.

*** The class definition only includes member variables and declaration of member functions**

calculator.cc contains the implementation of the member functions.

Problem 2 (50pt)

■Evaluation

- Addition, subtraction, multiplication, division, remainder, and exponentiation operations : 5 points each operations (5 * 6 = 30 points)
- assignment (=) operation : 10 points
- Print operation : 2.5 points
- Division by zero check : 2.5 points
- Wrong variable input check : 2.5 points
- Max number range check : 2.5 points
- 0 point if we cannot compile your program using the given makefile.

  * Your makefile should produce “calculator” executable file
- You must build a class named “calculator”, 0 point will be given otherwise.
- 0 point if we cannot compile your program using the given Makefile.
  - Your Makefile should produce executable file in your code directory. (++)
  - Executable file name should be “calculator” (++)
  - Don’t modify folder name. (0 point) (++)
  - You must build a class named “calculator”, 0 point will be given otherwise.