EEE 102 C++ Programming and Software Engineering II

Assessment 2
SDP Report

Student: Kai-Yu Lu
ID: 1614649
Date: 2018.4.16

Problem statement:

The first exercise for this assignment requires student to create another class called “iFraction” which is a sub-class from the base class “Fraction”. In addition, students should use iFraction class to show whether the fraction could be represented in the mixed fraction form. All the operators should be redefined by using operator overloading in class Fraction. As for the class iFraction, it should obtain a constructor to initialize the top and bottom. Besides, it needs have the function displaying the mixed fraction on screen. An external function called convertF should also be created to convert the improper fraction to mixed fraction. Finally, the program will contain following functions:

1. User can input numerator and denominator respectively (use space to
 separate them).
2. The program could perform operations (+, -, *, /) between two fractions.
3. The program could output the calculated fraction and compare the magnitude of two fractions. Decimal of the final will be also outputted.
4. The program could convert the improper fractions to mixed fractions.

Model Answer – Software Development Process

Analysis

Inputs:

- Two real numbers (minus sign is allowed) to represent numerator and denominator, use space to add next element.
- Input one operation: -, +, /, *.
- Back to the first step, create next fraction.

Outputs:

- Display instructions to let user know what to input.
- Display the calculated fraction and its mixed fraction.
- Find the greatest common divisor and then let it divide the top and the bottom.
- Output the information to tell user which magnitude of fractions is larger.

Design:

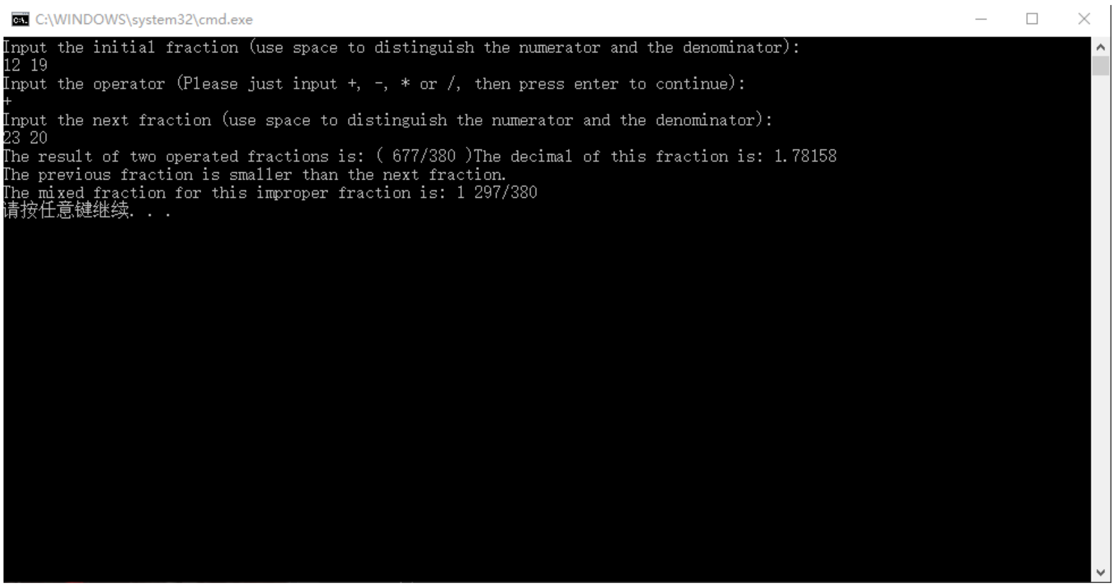
1. Create class and initialize the top and the bottom. If bottom is negative, set the bottom to be positive and top to be negative, of which aim is to standardize the format of the fraction.
2. Use operator overloading to rewrite the operators (+, -, *, /).

3. Write functions to input and output.
4. Use formula to compare the magnitudes between two fractions.
5. Create menu to let user choose the operation.
6. Convert fraction to decimal.
7. Create the sub-class iFraction from class Fraction and initialize the numerator and denominator. In class iFraction, a method to display the mixed fraction should be created.
8. Design a function called convertF which is the friend function of two other classes.

Implementation:

See the code for exercise 1 in the file: exercise 1.cpp

Testing:



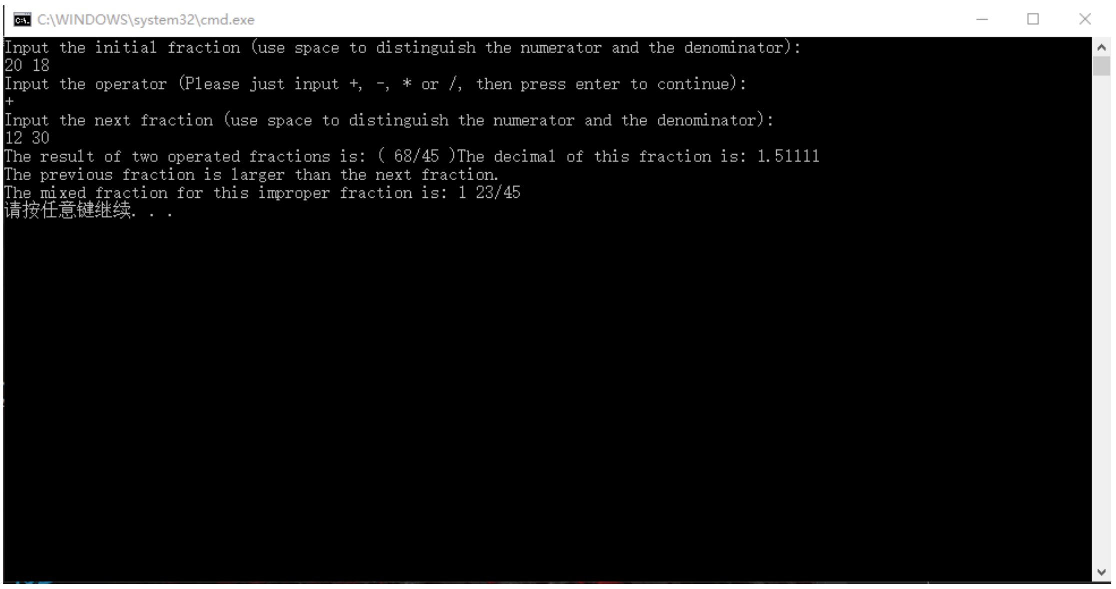
```
Input the initial fraction (use space to distinguish the numerator and the denominator):
12 19
Input the operator (Please just input +, -, * or /, then press enter to continue):
+
Input the next fraction (use space to distinguish the numerator and the denominator):
23 20
The result of two operated fractions is: ( 677/380 )The decimal of this fraction is: 1.78158
The previous fraction is smaller than the next fraction.
The mixed fraction for this improper fraction is: 1 297/380
请按任意键继续. . .
```

Figure 1: The operation is "+" and the first fraction is smaller than

the next fraction. The mixed fraction has “1” carry.

Figure 1 could prove following functions work:

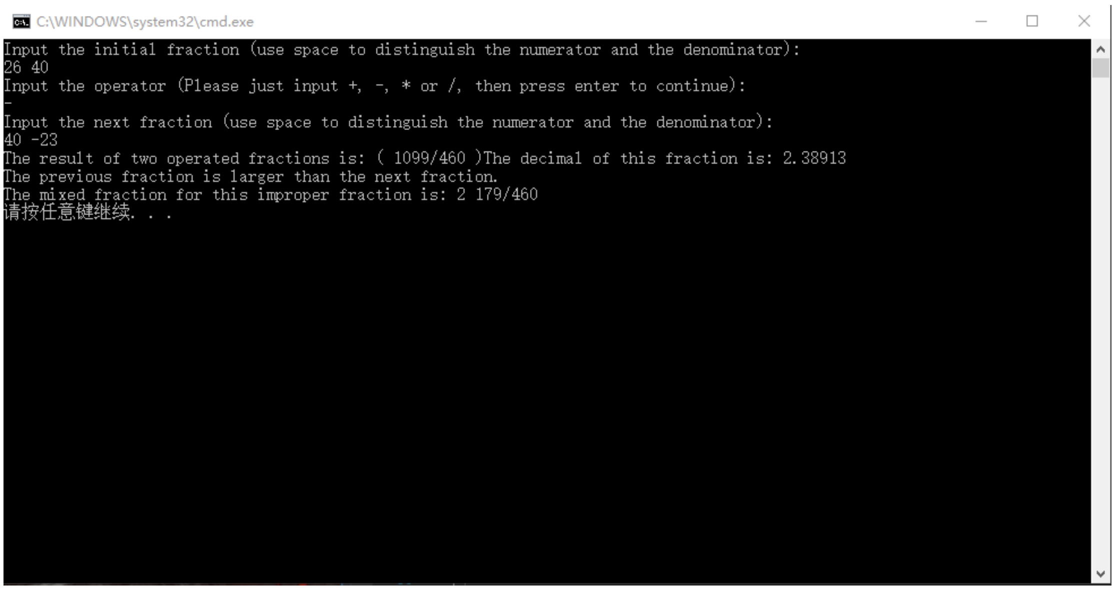
1. Operator “+”.
2. Convert fraction to decimal.
3. The function of picking out which fraction is smaller.
4. The function to convert improper fraction to mixed fraction.



```
C:\WINDOWS\system32\cmd.exe
Input the initial fraction (use space to distinguish the numerator and the denominator):
20 18
Input the operator (Please just input +, -, * or /, then press enter to continue):
+
Input the next fraction (use space to distinguish the numerator and the denominator):
12 30
The result of two operated fractions is: ( 68/45 )The decimal of this fraction is: 1.51111
The previous fraction is larger than the next fraction.
The mixed fraction for this improper fraction is: 1 23/45
请按任意键继续. . .
```

Figure 2: The operation is “+” and the first fraction is larger than the next fraction.

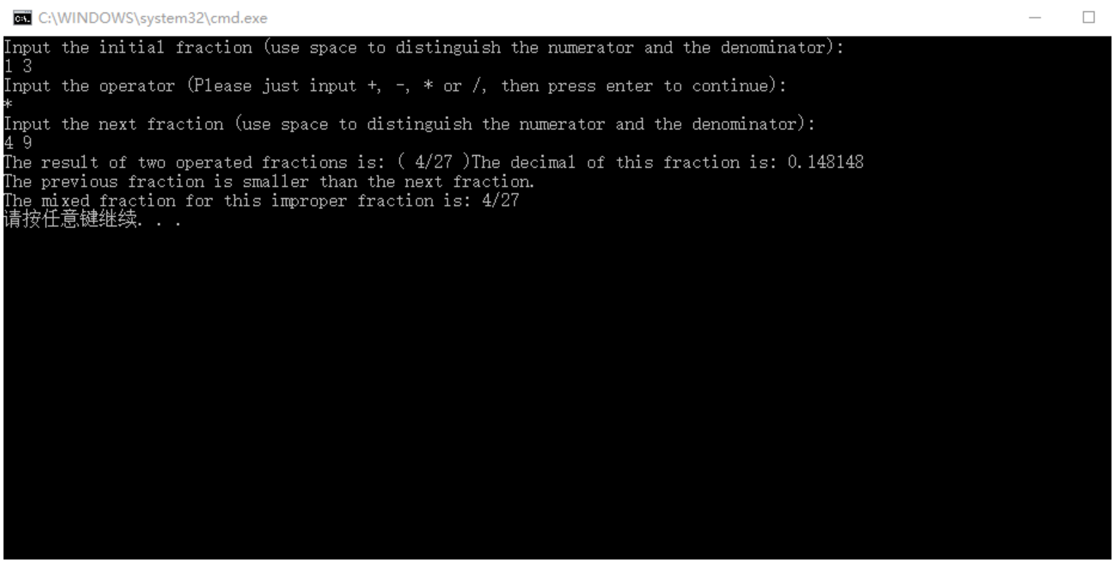
Figure 2 shows that the program could identify which fraction is larger.



```
Input the initial fraction (use space to distinguish the numerator and the denominator):
26 40
Input the operator (Please just input +, -, * or /, then press enter to continue):
-
Input the next fraction (use space to distinguish the numerator and the denominator):
40 -23
The result of two operated fractions is: ( 1099/460 )The decimal of this fraction is: 2.38913
The previous fraction is larger than the next fraction.
The mixed fraction for this improper fraction is: 2 179/460
请按任意键继续. . .
```

Figure 3: The operation is "-"

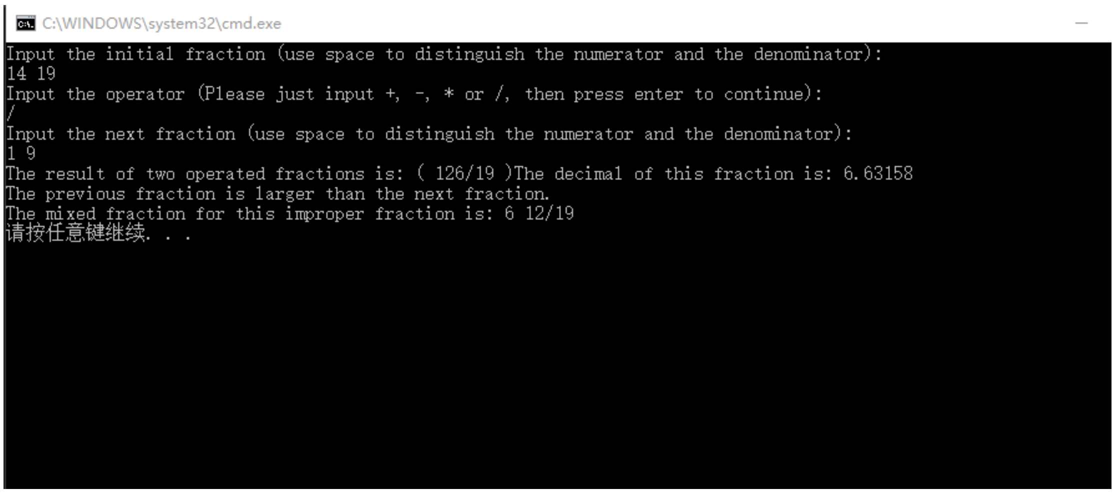
Figure 3 shows that the function of operator "-" work.



```
Input the initial fraction (use space to distinguish the numerator and the denominator):
1 3
Input the operator (Please just input +, -, * or /, then press enter to continue):
*
Input the next fraction (use space to distinguish the numerator and the denominator):
4 9
The result of two operated fractions is: ( 4/27 )The decimal of this fraction is: 0.148148
The previous fraction is smaller than the next fraction.
The mixed fraction for this improper fraction is: 4/27
请按任意键继续. . .
```

Figure 4: The operation is "*".

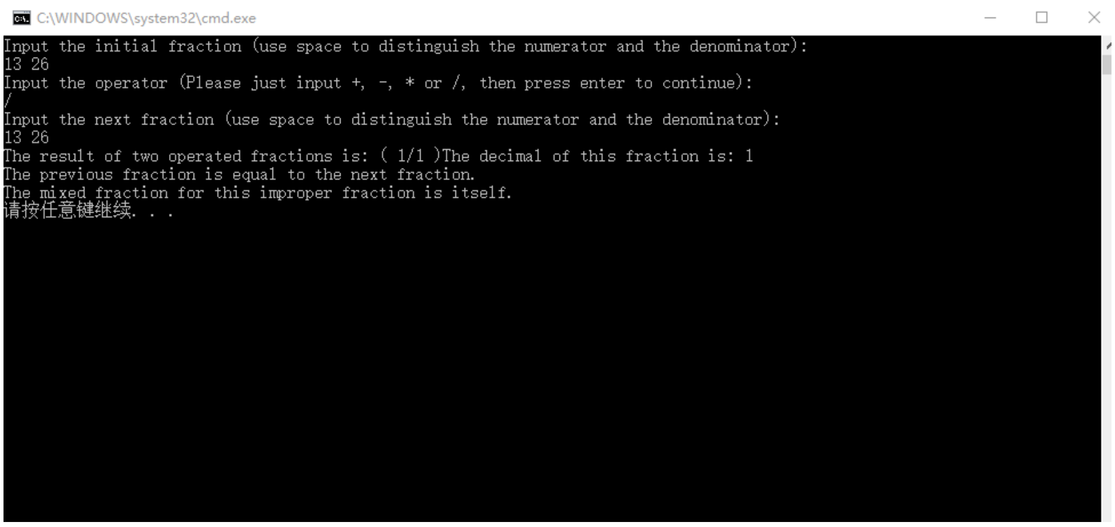
Figure 4 shows that the function of operator "*" work.



```
Input the initial fraction (use space to distinguish the numerator and the denominator):
14 19
Input the operator (Please just input +, -, * or /, then press enter to continue):
/
Input the next fraction (use space to distinguish the numerator and the denominator):
1 9
The result of two operated fractions is: ( 126/19 )The decimal of this fraction is: 6.63158
The previous fraction is larger than the next fraction.
The mixed fraction for this improper fraction is: 6 12/19
请按任意键继续. . .
```

Figure 5: The operation is “/”.

Figure 5 shows that the function “/” work.



```
Input the initial fraction (use space to distinguish the numerator and the denominator):
13 26
Input the operator (Please just input +, -, * or /, then press enter to continue):
/
Input the next fraction (use space to distinguish the numerator and the denominator):
13 26
The result of two operated fractions is: ( 1/1 )The decimal of this fraction is: 1
The previous fraction is equal to the next fraction.
The mixed fraction for this improper fraction is itself.
请按任意键继续. . .
```

Figure 6: The condition that if two fractions are equal.

Figure 6 shows that if the final fraction is equal to 1, the mixed fraction is itself and two equivalent fractions could be also compared.

In conclusion, these tests could prove most functions are effectively achieved, which are:

1. Operator overloading (+, -, *, /).

2. Comparison.
3. Convert fraction to decimal.
4. Convert improper fraction to mixed fraction.
5. Corresponding output functions.