1. **Problem statement:**

To derive a subclass iFraction from the base class Fraction. The class iFraction can represent a mixed fraction on screen. It should detect whether the denominator is zero and ask user to enter again. Additionally, it should detect illegal input such as characters and ask user to enter the legal integer number. When user enters a fraction without reduction, it should help user to do the reduction. When user enters an improper fraction, the program should convert it into a mixed fraction. The relationship between the carry and fraction should be addition, which is designed for “a negative fraction” case. The mixed fractions can be converted into improper fractions by an external function.

1. **Analysis:**

Inputs: three integer numbers representing carry, numerator and denominator respectively of a mixed fraction. The numerator and denominator are inherited from protected members of base class Fraction. The carry is defined as private member in subclass iFraction.

Outputs: the mixed fraction and the converted improper fraction. For mixed fraction, the output is still three integers with “/”. For improper fraction, the output is two integers with”/”.

Additional requirements: must derive a subclass iFraction from the base class Fraction

1. **Design:**

Class definition

1. Define a subclass iFraction publicly inherited from base Fraction
2. Declare a friend function convertF which is meant to convert mixed fractions to improper fractions
3. Define a default constructor, a normal constructor and a destructor.
4. Define four member functions which have different responsibilities of converting improper fractions to mixed fractions, displaying, inputting and reduction.
5. For member function convert, give the number of numerator divided by denominator to carry and let top decrease by the value of carry multiplied by denominator
6. For member function display, display the fraction based on its numerator and denominator. If carry equals to zero, do the reduction and display the fraction directly by using output function from base class Fraction. If numerator is equal to denominator, display “carry plus one”. If numerator is less than denominator, do the reduction and output the carry and the fraction. If numerator is larger than denominator, convert it into mixed fraction firstly, do the reduction and then output the carry and the fraction.
7. For member function input, ask user to enter the carry, numerator and denominator one by one.
8. For member function reduction, first check whether the denominator is less than zero and reverse it if yes. Next, find the most common divisor and let the numerator and denominator divided by it.

Main function

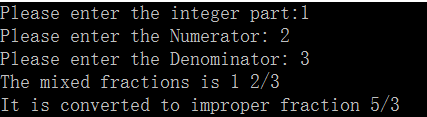
1. Declare an object A of class iFraction and an object B of class Fraction
2. Ask user to enter an mixed fraction and store it as A
3. Display the mixed fraction the user entered
4. Convert the mixed fraction to improper fraction and store it as B
5. Display the improper fraction B

**4. Implementation**: see C++ code in file 1405347\_4-1.cpp, 1405347\_fraction.h and 1405347\_fraction.cpp with comments.

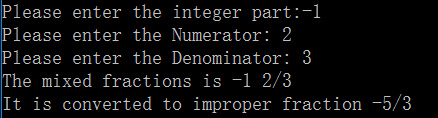
**5. Testing:**

The C++ program was tested by carrying out a set of experiments and the C ++program output was verified successfully. For example,

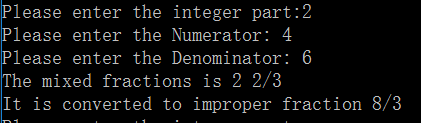
When input a normal mixed fraction, the result is



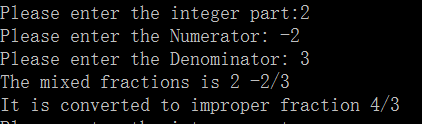
When input a mixed fraction with a negative carry, the result is



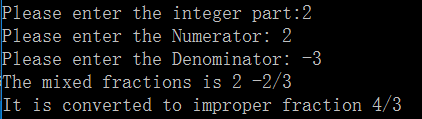
When input a mixed fraction without reduction, the result is



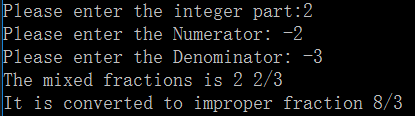
When input a mixed fraction with a negative numerator, the result is



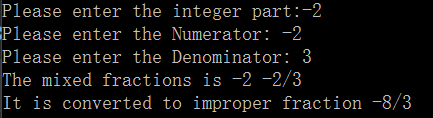
When input a mixed fraction with a negative denominator, the result is



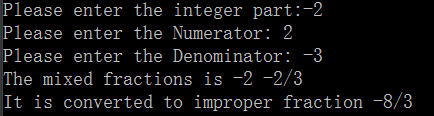
When input a mixed fraction with both negative numerator and denominator, the result is,



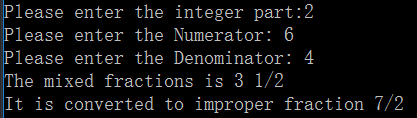
When input a negative integer and a negative numerator, the result is



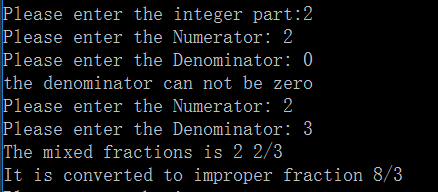
When input a negative integer and a negative denominator, the result is



When input an improper fraction, the result is



When input the denominator as zero, the program will tell wrong and ask user to enter again.



When input characters, the program will tell wrong and ask user to enter again.

