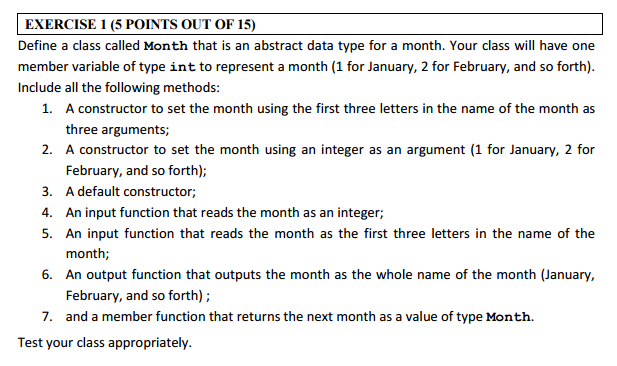
**Assignment 2**

**1301058**

**Zhang Junming**

**Exercise 1**

Question



Model Answer

Software Development Process

1. **Problem statement**

Define a class which called Month to write a program, this program could construct month by their first three letters or by an integer. User could choose how to get the full name of the month, they could enter the integer or the first three letters of the month.

1. **Analysis**

Inputs:

1. An integer
2. First three letters of month

Outputs:

1. Full name of the month

Additional requirements or constraint

Just one member variable of type **int** to represent a month.

**3. Design**

Algorithm

1. Adding “iostream” and “string” header files.
2. Using of the ste namespace.
3. Define a class which called Month

Private:

int month – represent a month

Public:

Month() - default constructor

<1> let month equal to 0(represent January).

void SetMonthOf3Letters(string l1,stringl2,stringl3) - a constructor to set month by their first three letters.

<1> Declare a string l and then store 12 first three letters of the month in it.

<2> Declare a string l4 equal to l1 plus l2 and l3 to get the first three letters of the month.

<3> Setting up a loop to confirm which months match the first three letters.

void SetMonthOfInt(int Month) - a constructor to set month by an integer.

<1> let month equal to Month minus 1 because 0 represent January.

void InputAndReadMonthAsInt() - user could input a integer and then the program will return the full name of the month with the integer.

<1> Ask user to enter an integer.

<2> declare and store the value in n.

<3> let month equal to n minus 1.

void InputAndReadMonthAs3Letters() user could input the first three letters of the month and then the program will return the full name of the month with the integer.

<1> Declare a string l and then store 12 first three letters of the month in it.

<2> Declare 4 stings to store first three letters and the letters which they connect together.

<3> Setting up a loop to ask user enter the letter, if they enter error, let user enter again.

<4> Connect the first three letters.

<5> Setting up a loop to confirm which months match the first three letters.

<6> if cannot find the related month, give the answer to user.

void OutputMonth() - program will display the full name of the month on the screen.

<1> Declare a string Fullname and then store the full name of 12 month in it.

<2> Output the full name of month based on the integer in the class member.

<3> If the value of month not in the area between 0 and 11(January to December), tell user check their input.

Month next() – program will return the value of the next month.

<1> Month t - construct a class t to store related information

<2> To judge the integer of month, if the value of the integer is 11(represents December) then return the 0(represents January) to month, else the value of integer will plus 1(represents next month).

<3> Return the value of integer to t.

1. Write the main function

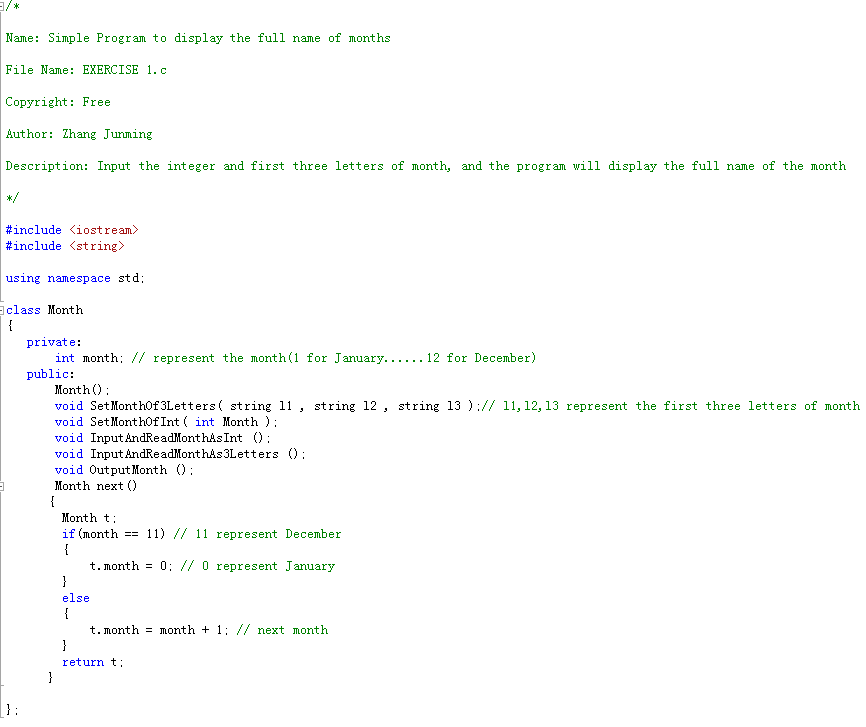
<1> Display the task menu to user.

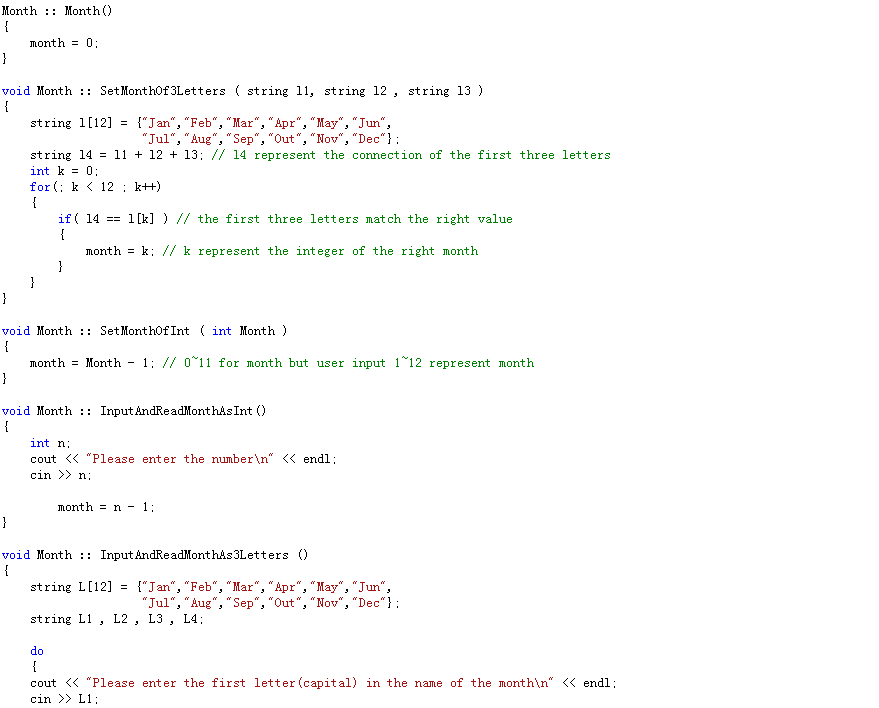
<2> Using switch and do-while function to store the different operations.

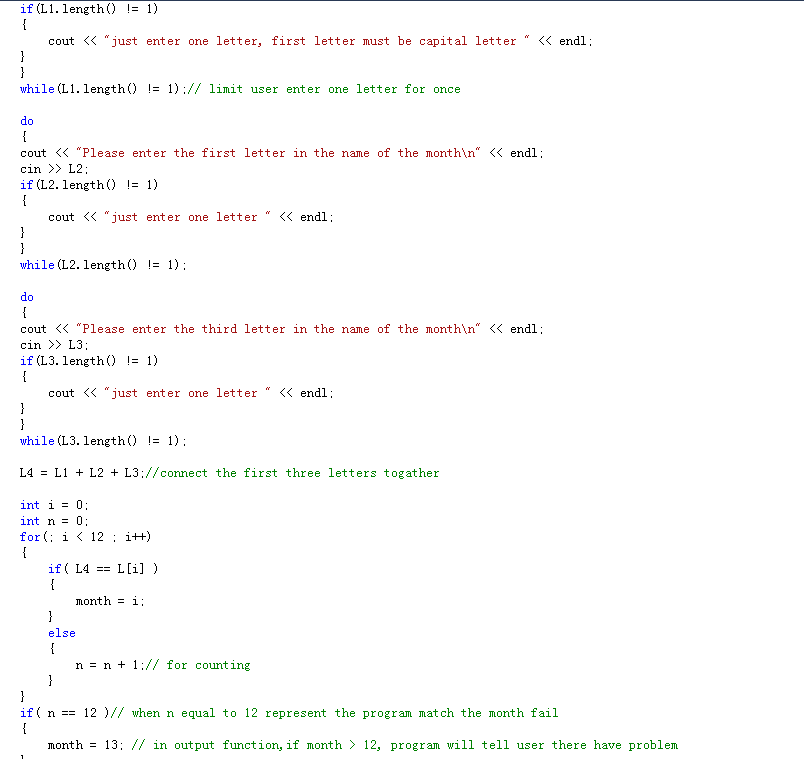
<3> test data was input in the program, therefore the default construct is January and the integer construct is October.

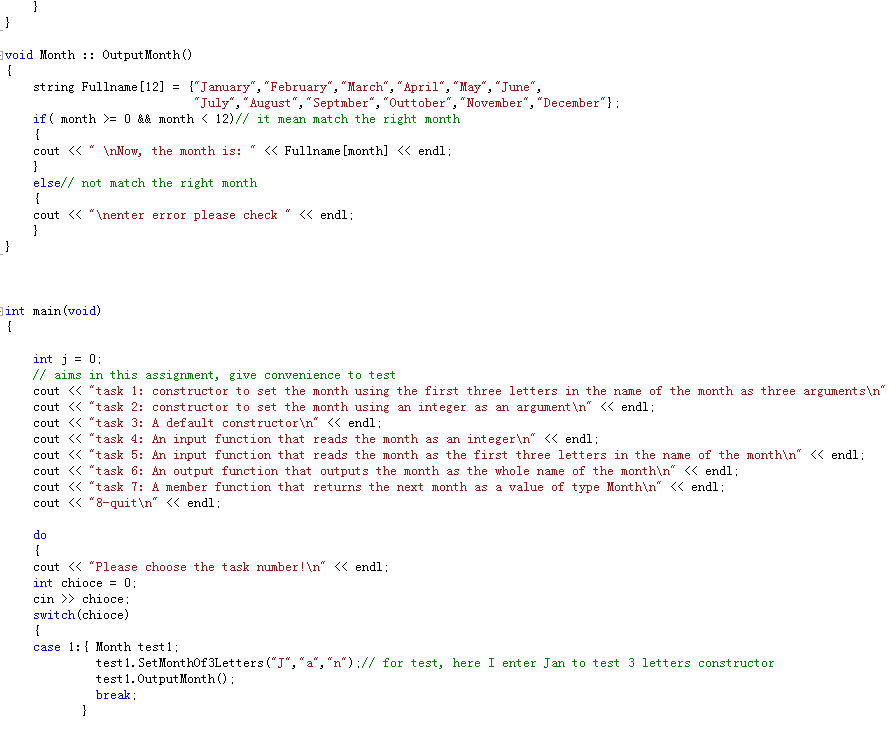
1. **Implementation**:

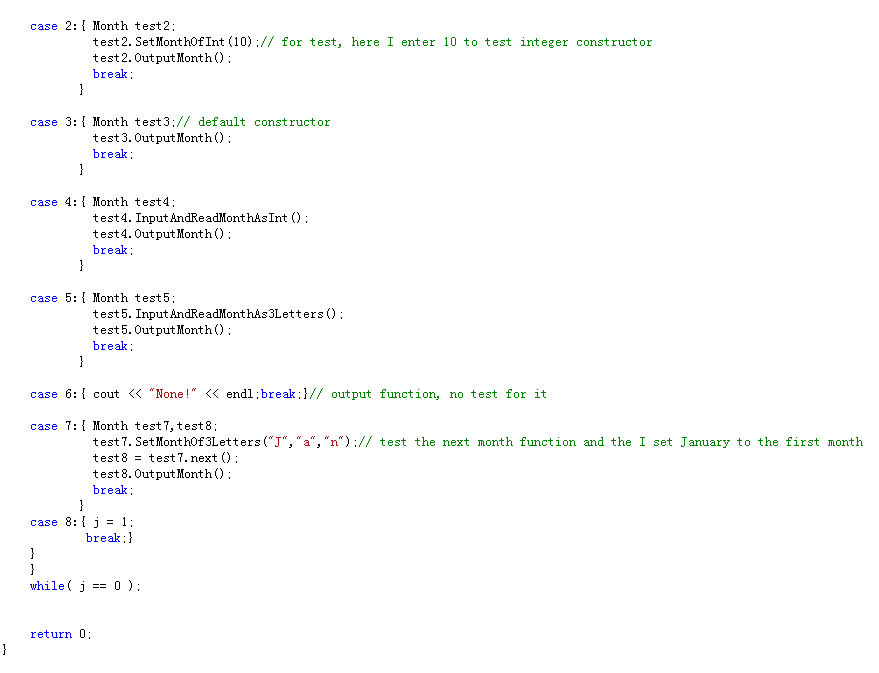
See the C code in file exercise1.c with comments.



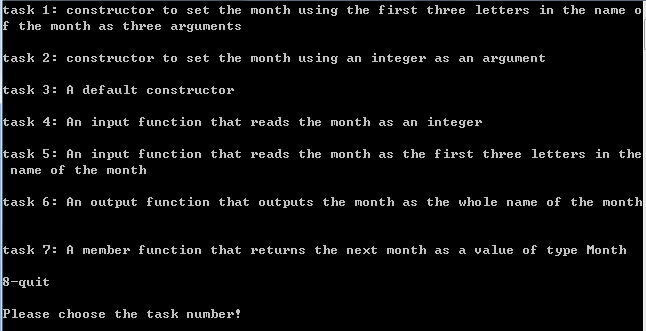




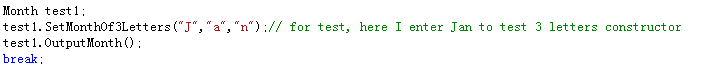


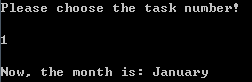


1. **Testing:**

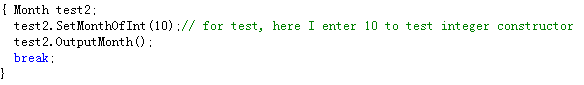


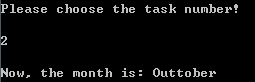
Task menu





Task 1

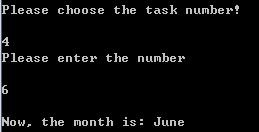




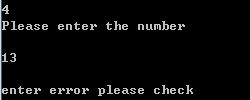
Task 2



Task 3: default constructor is January

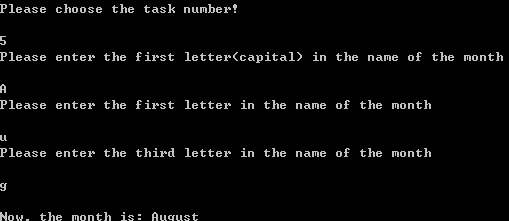


Enter correct

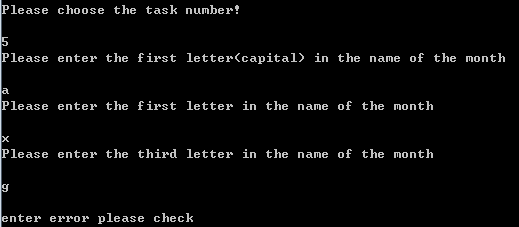


Enter wrong

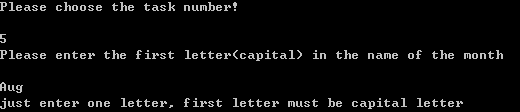
Task 4



Enter correct

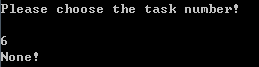


Enter wrong

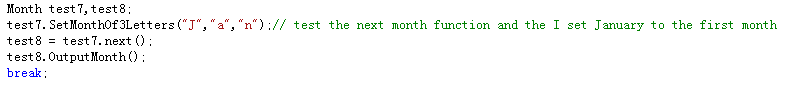


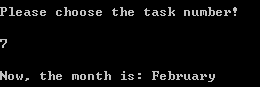
Enter wrong

Task 5



Task 6: output function, display in other task

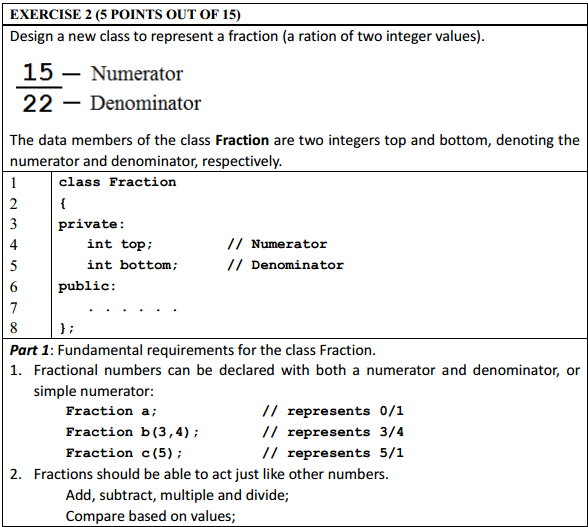


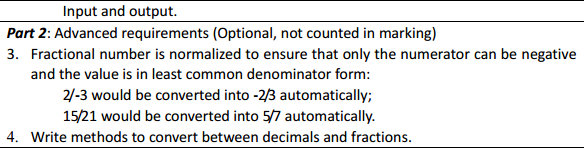


Task 7: next month function and input January for the first month

**Exercise 2**

Question





Model Answer

Software Development Process

**1**. **Problem statement**

Write a program to construct fraction and the fraction could add, subtract, multiple and divide.

**2. Analysis**

Inputs:

The value of fraction (denominator and numerator).

Outputs:

The result of tow fractions’ add, subtract, multiple, divide and compare.

Additional requirements or constraint

Only top and bottom in the class member.

**3. Design**

Algorithm

1. Adding “iostream” header files.

2. Using of the ste namespace.

3. Define a class which called Fraction

Private

int top – represent the numerator

int bottom – represent denominator

Public

Fracion() – default constructor

<1> let top equal to 0 and bottom equal to 1

Fraction(int N,intD=1) – normal constructor

<2> let top equal to N and bottom equal to D

void Output()- display the task result on screen

void output() – display the calculate result on screen

Fraction operator+(const Fraction & n) – re-define symbol+

<1> construct a new fraction t to receive data

<2> reduction of fractions to a common denominator

<3> calculate the value of top and bottom

<3> return the value of top and bottom

Fraction operator-(const Fraction & n) – re-define symbol

The same as operator+

Fraction operator\*(const Fraction & n) – re-define symbol\*

<1> construct a new fraction t to receive data

<2> the result of numerator equal to two numerators’ multiple

<3> the result of denominator equal to two denominators’ multiple

Fraction operator/(const Fraction & n) – re-define symbol/

The same as operator\*

Fraction operator<(const Fraction & n) – re-define symbol<, could compare two fraction’s value

<1> construct two new fractions to receive data

<2> reduction of fractions to a common denominator

<3> compare the value of numerator

<4> display the result

4. Write main function

<1> display the menu

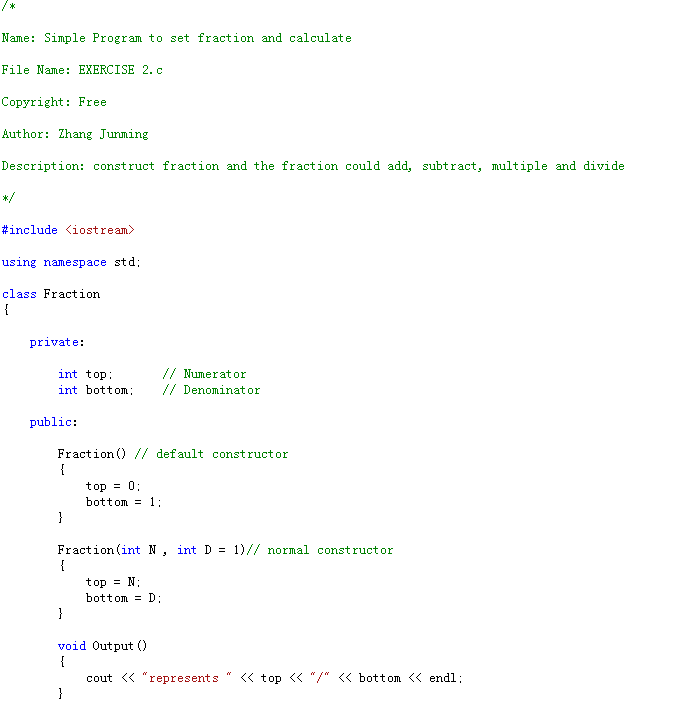
<2> setting up a loop to let user choose function

<3> show tasks in one option

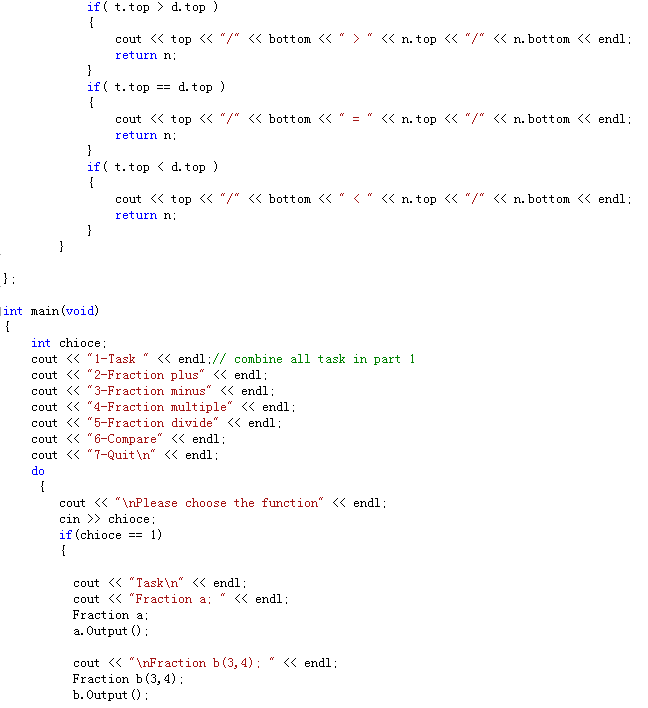
<4> add the calculate function to other options

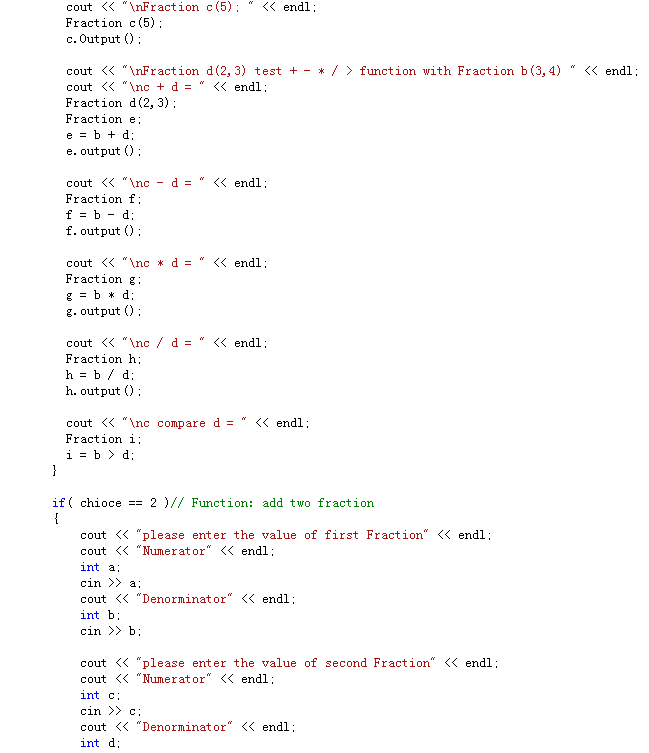
**4. Implementation**:

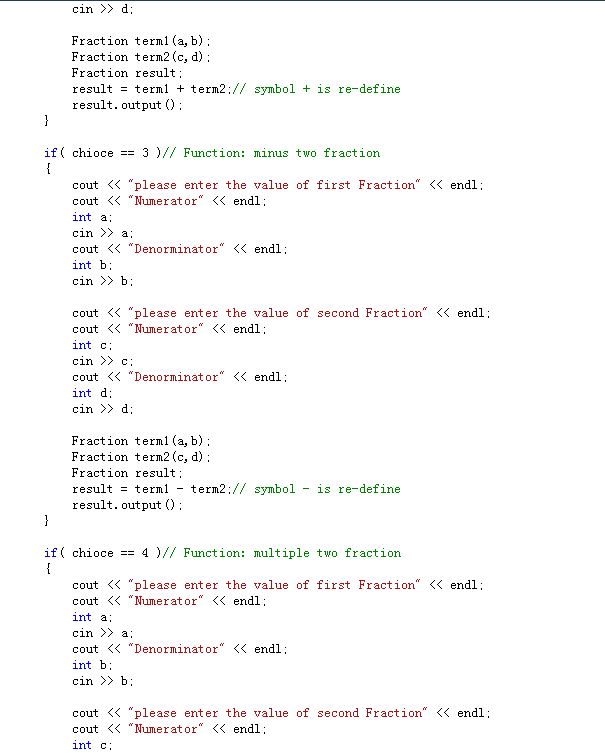
See the C code in file exercise1.c with comments.

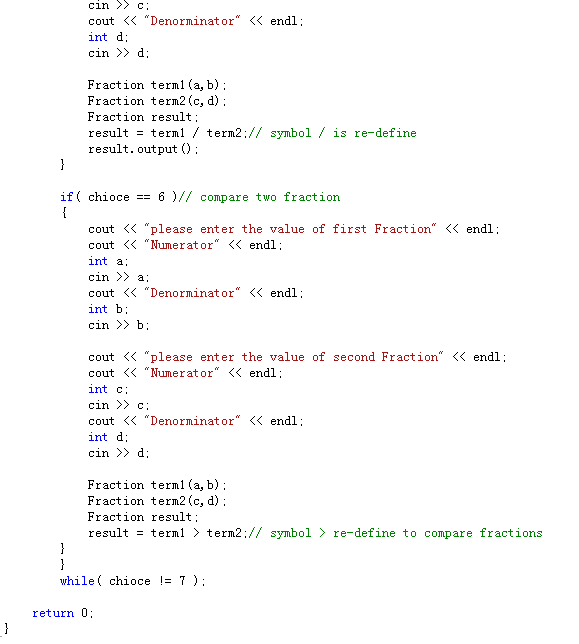




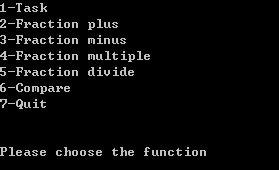




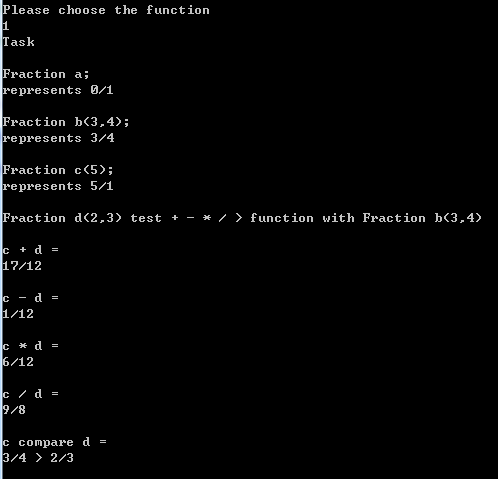
****

****

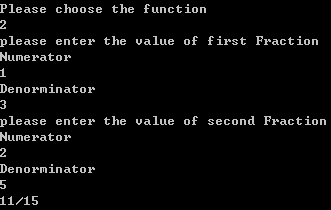
**5. Testing:**



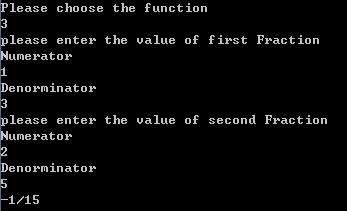
Menu



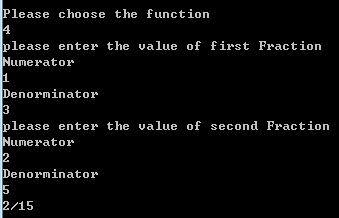
Require of exercise 2



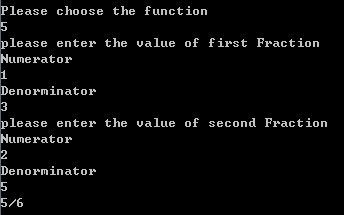
Add function



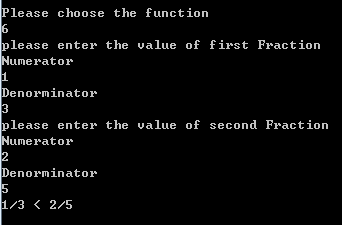
Subtract function



Multiple function



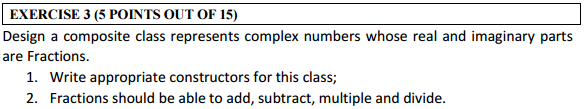
Divide function



Compare function

**Exercise 3**

Question



Model Answer

Software Development Process

**1**. **Problem statement**

Write a program using composite class to calculate complex number’s arithmetic.

1. **Analysis**

Inputs:

Two complex numbers’ real part and imaginary part using fraction

Outputs:

The arithmetic result of two complex numbers

Additional requirements or constraint

Using composite class

**3. Design**

Algorithm

1. Adding “iostream” header files.

2. Using of the ste namespace.

3. Define a class which called Fraction

Private

int top – represent the numerator

int bottom – represent denominator

Public

Fracion(int N=0,int D=0) – normal constructor

<1> let top equal to 0 and bottom equal to 0

int Gettop() – to get the value of numerator

<1> return top

int Getbottom() – to get the value of denominator

<1> return bottom

Fraction Fadd (Fraction n) – Operator overloading Fadd

<1> construct a new Fraction t to receive data

<2> reduction of fractions to a common denominator

<3> calculate the value of top and bottom

<3> return the value of top and bottom

Fraction Fminus (Fraction n) – Operator overloading Fminus

The same as Fadd

Fraction Ftimes (Fraction n) – Operator overloading Ftimes

<1> construct a fraction t to receive data

<2> the result of numerator equal to two numerators’ multiple

<3> the result of denominator equal to two denominators’ multiple

Fraction Fdivision (Fraction n) – Operator overloading Fdivison

The same as Ftimes

4. Define a class which called Fraction

Private

Fraction real – represent the fraction real part

Fraction imag – represent the fraction imaginary part

Public

complexClass() – default constructor

complexClass(Fraction r,Fraction i):reak®,imag(i){}) – normal composite constructor

complexClass Csum (complexClass n) – Operator overloading Csum

<1> construct a new complex t to receive data

<2> the sum of real part equal to the sum of two complex numbers’ fraction real part

<3> the sum of imaginary part equal to the sum of two complex numbers’ fraction imageinary part

<4> return the value of real and imag

complexClass Cminus (complexClass n) – Operator overloading Cminus

The same as Cminus

complexClass Ctimes (complexClass n) – Operator overloading Ctimes

<1> construct a new complex t to receive data

<2> multiple formula : A\*B=ac-bd+(ad+bc)i

<3> return the value of real and imag

Fraction Fdivision (Fraction n) – Operator overloading Fdivison

<1> construct a new complex t to receive data

<2> division formula : (a+bi)/(c+di)   
  =(a+bi)\*(c-di)/(c+di)\*(c-di)   
 =(ac+adi+bci+bd)/(c\*c+d\*d)

<3> return the value of real and imag

void output() – display the calculate result

5. Write main function

<1> display the menu

1 for complex number plus;

2 for complex number minus;

3 for complex number multiple;

4 for complex number divide

5 for quit

<2> setting up a loop to ask user choose function

<3> complex plus:

1. ask user input the value of two complex number’s fraction.

2. construct the related complex number

3. calculate the sum of two complex numbers

4. display the result

<4> complex minus:

1. ask user input the value of two complex number’s fraction.

2. construct the related complex number

3. calculate the sum of two complex numbers

4. display the result

<5> complex multiple:

1. ask user input the value of two complex number’s fraction.

2. construct the related complex number

3. calculate the sum of two complex numbers

4. display the result

<6> complex divide:

1. ask user input the value of two complex number’s fraction.

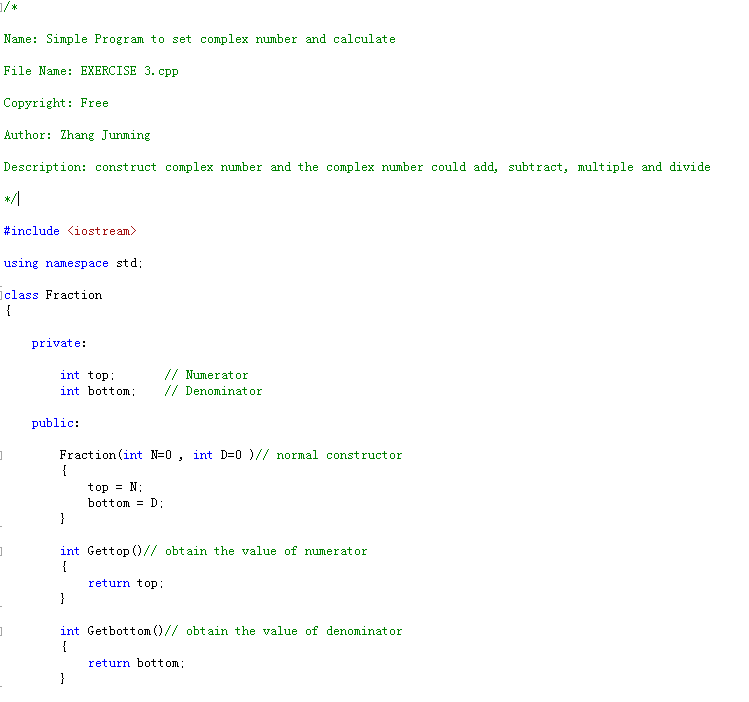
2. construct the related complex number

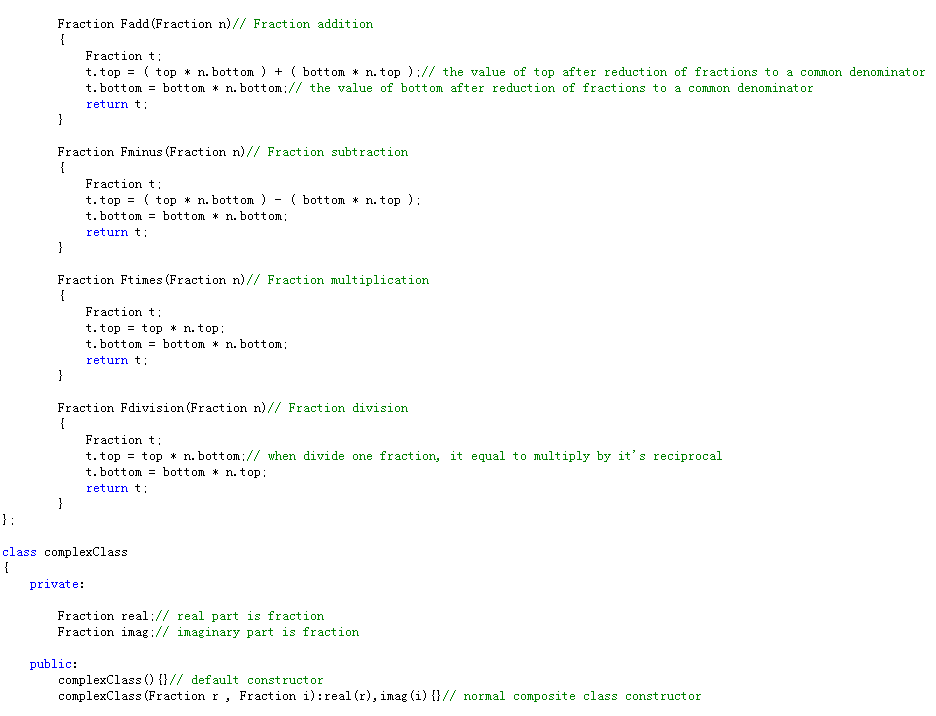
3. calculate the sum of two complex numbers

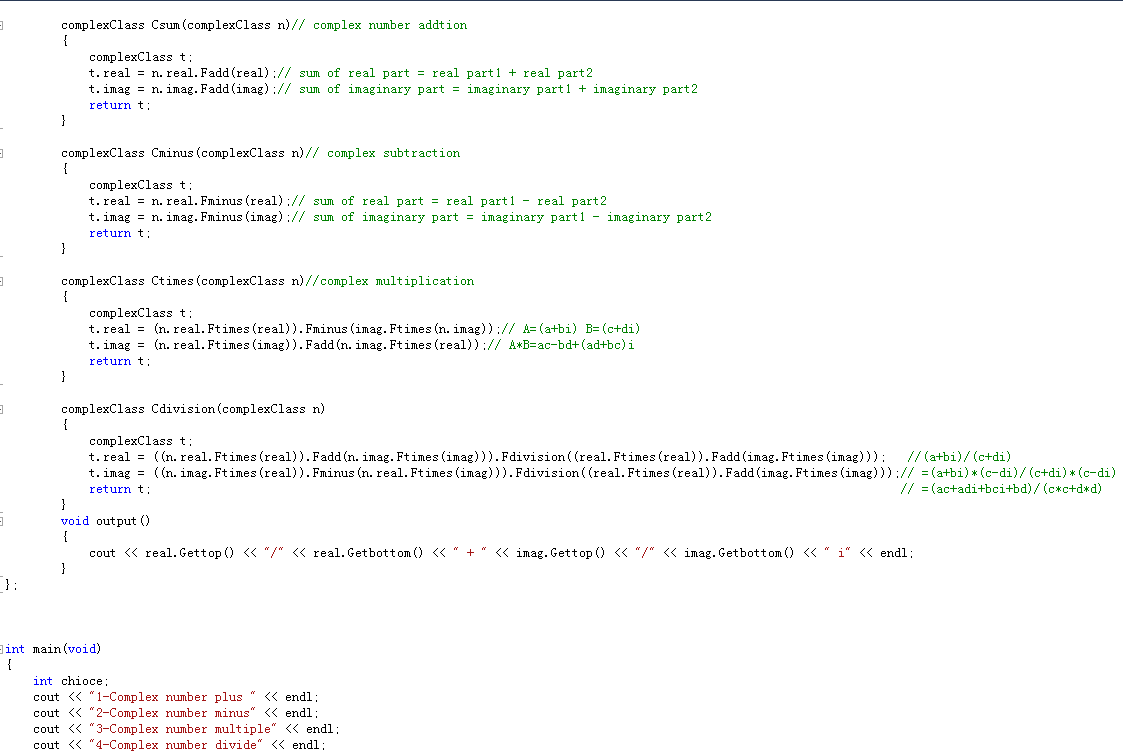
4. display the result

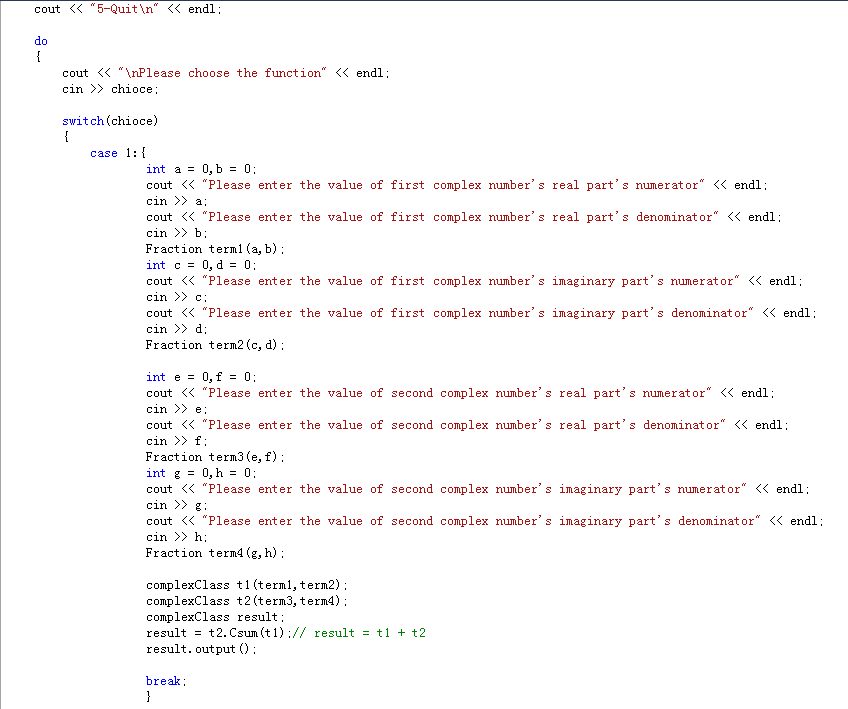
**4. Implementation**:

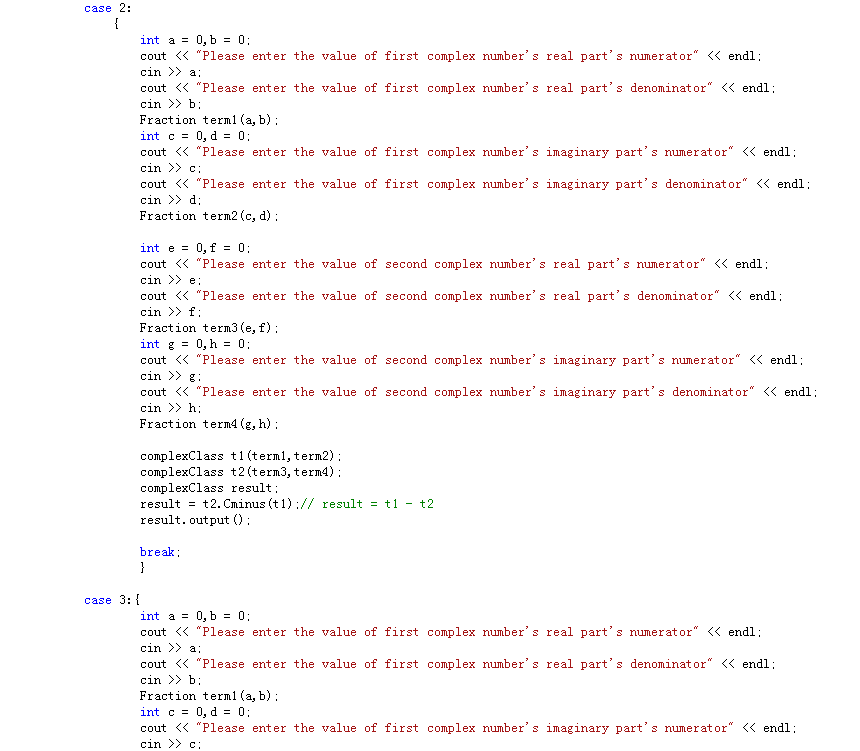
See the C code in file exercise1.c with comments.

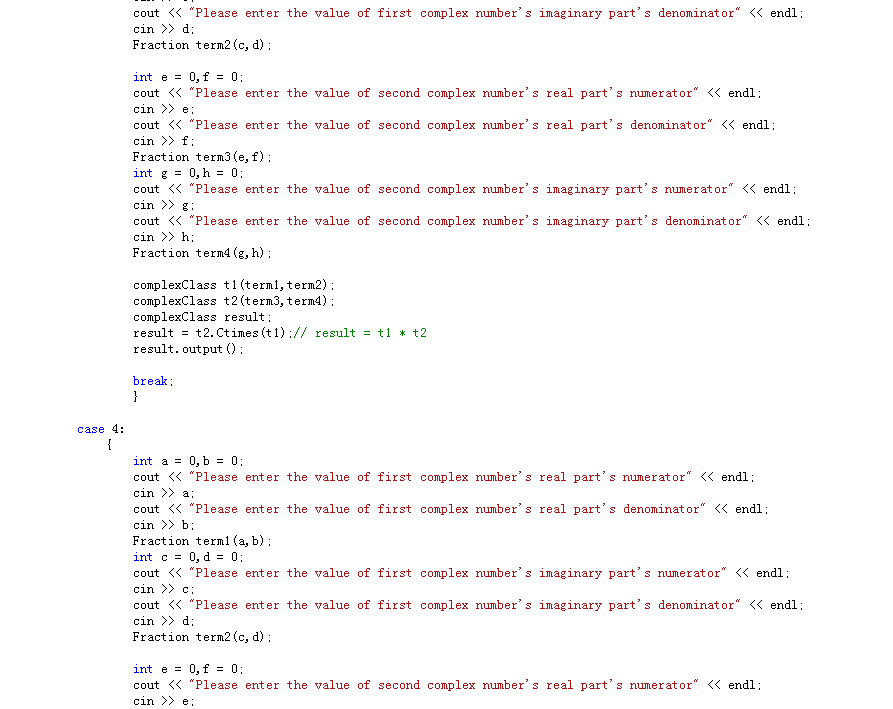


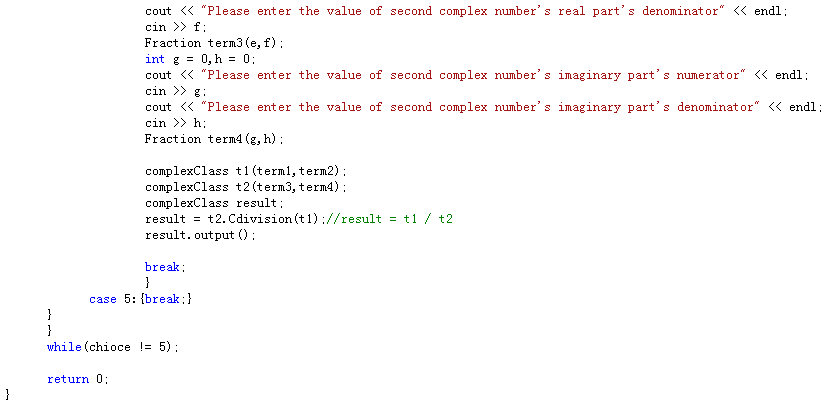




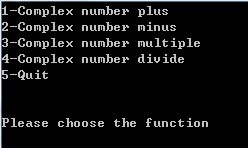




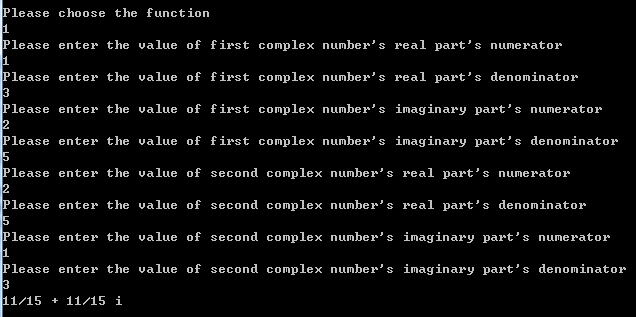




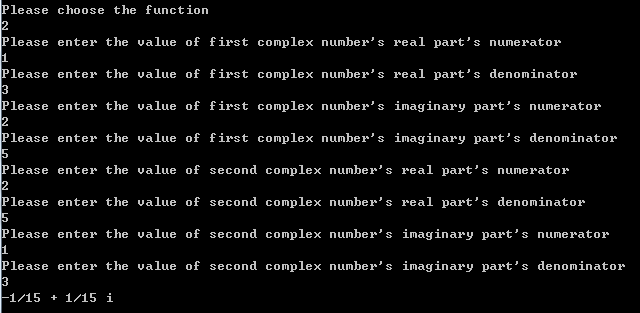
**5.Testing:**



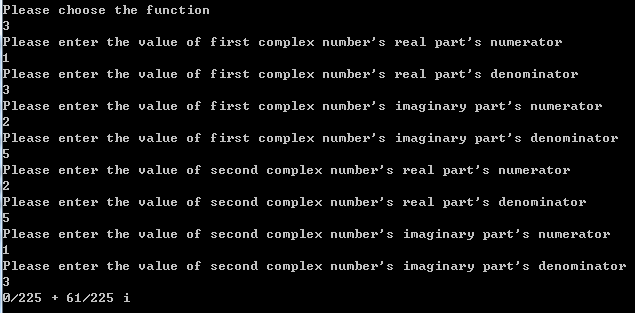
Menu



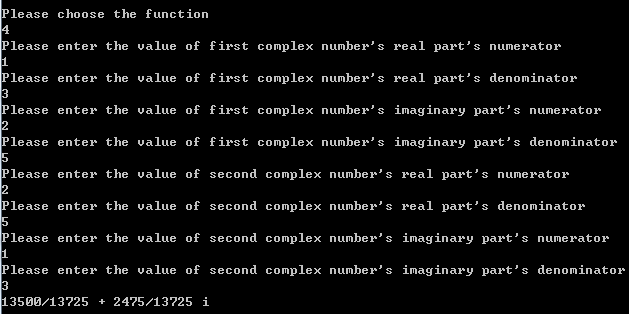
Complex number addition



Complex number subtraction



Complex number multiplication



Complex number division