Xi'an Jiaotong-Liverpool University
西交利物浦大学

Department of Electrical and Electronic Engineering

CPT106(S2) C++ Programming and Software Eng. II

CPT106 C++ Programming Report

Name :Tiankuo.Jiao

Student ID :1929098

Catalogue

1.Specification（Problem statement）

1.1 For exercise1

1.1.1 Description of tasks and functions

In this experiment, we need to create a function, which can check whether two integer arrays have the same elements, ignoring the order and multiplicities.At the same time, when we input two arrays, we also need to input the number of integers(len) contained in the two arrays into the program, the specific part of the input is as follows:

1) int a[ ]

2) int lenA

3) int b[ ]

4) int lenB

In the main function, the program need to print out the "same" if the function returns true; print out "different" if the function returns false.Despite the requirement considers that the two arrays should be of different lengths, my program can recognize arrays of the same and different lengths.

1.1.2 Error analysis and solutions

1.1.2.1 Error

At the same time, if the user input integers wrongly, the program will report an error information and allow the user to enter again.The error is as follows:

1) User input special characters as elements of arrays, such as: @

2) User input letter as a element of arrays, for example: a

3) The array length entered by the user does not match the number of characters entered

4) When the user enters the length of the array, a non-number is entered.

5)When the user enters the length of the array, a special character is entered like @

1.2.2.2 Solutions

1) When the user enters the wrong character, we need to report the error and ask the users to type again

2) When the characters entered by the user do not match to the defined array length, the system will automatically select the characters according to the array length

1.2 For exercise 2

1.2.1 Description of tasks and functions

In this exercise, we need to design a new class to represent a fraction (the ratio of two integer values). Fractions need to be declared in terms of a numerator, denominator, or simple numerator, for example:

1) Fraction a; represents 0/1

2) Fraction b(3,4); represents 3/4

3) Fraction c(5); represents 5/1

Also, we need to implement the following function in main:

1. Fractions can be added, subtracted, multiplied and divided, just like any other number. We can judge that the first fraction is larger by subtracting the two fractions if the final result is positive.

For example:

void add(Fraction a);

void sub(Fraction a);

void mul(Fraction a);

void div(Fraction a);

2. Input and output

The following is advanced requirements

3.If the denominator has a minus sign in the numerator, we need to put the minus sign in the numerator.

For example:

2/-3 is automatically converted to -2/3

4. If we can simplify the fraction, we need to show what the simplified fraction is.

For example:

15/21 would be converted into 5/7 automatically

5. Fractions can be converted to decimals

1.2.2 Error analysis and solutions

1.2.2.1 Error

Obviously, if the user inputs an incorrect character, we need to prompt the user for an error and ask the user to retype, which can happen as follows:

1)When the users typed in the denominator, they typed in 0

2) When selecting the function, the user entered characters outside the function, including @ and A

3) When users enter the numerator and denominator, you enter a non-number. For example:!

4)When the second time to add, subtract, multiply and divide operation, users input wrongly.

1.2.2.2 Solutions

1) If the denominator of user input is detected 0 , the system will tell the user the input error and ask the user to re-enter

2) When selecting a function, if the wrong number is entered, the system will report an error

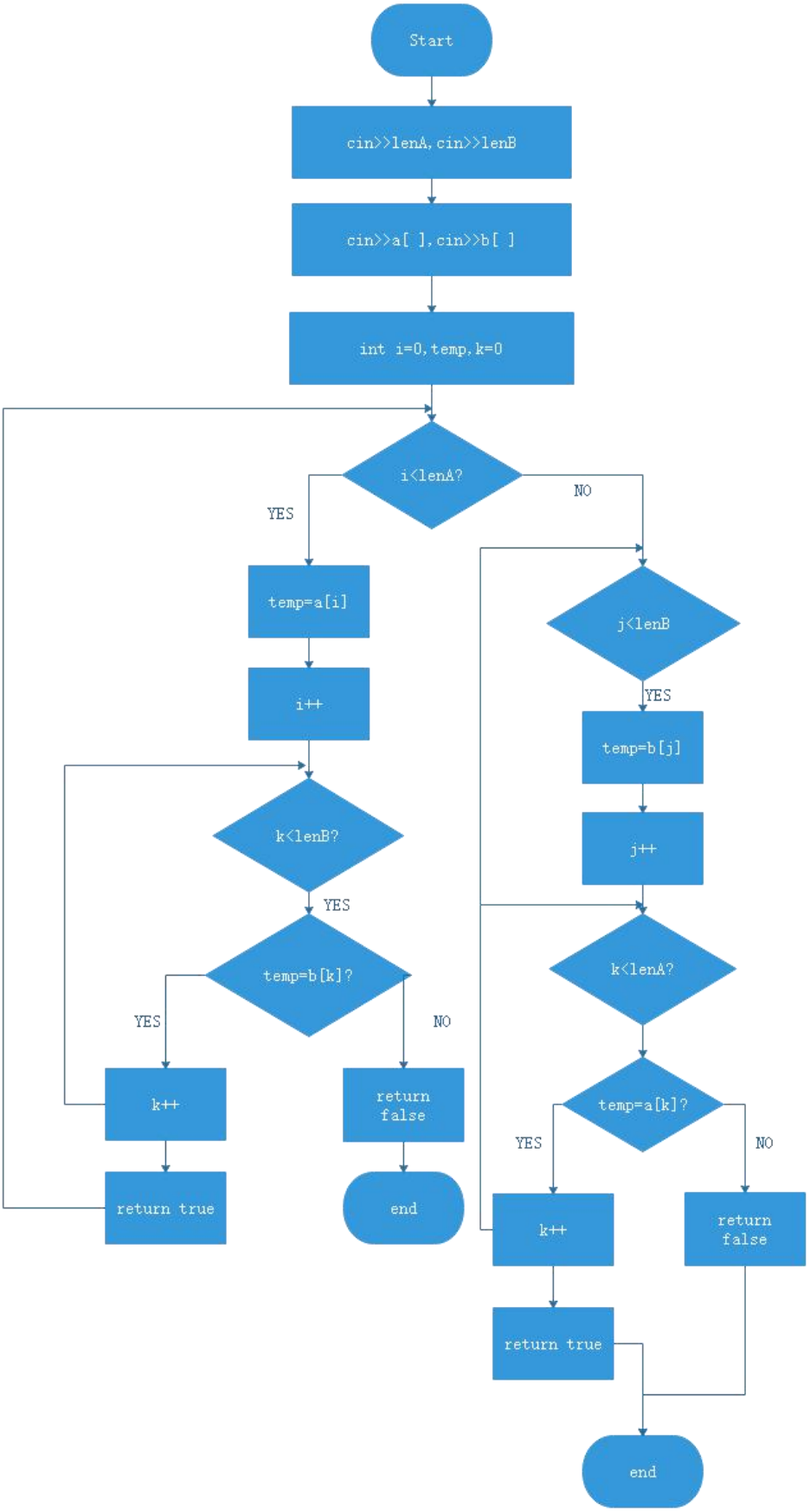
3) When entering numerator and denominator, if the system detects the user's input is not a number, the system will prompt the user for input error and ask the user to enter again

2.Analysis

2.1 For exercise1

We define two for loops in the same_Array function. The first for loop determines whether an element in a[ ] exists in b[ ]. The second for loop determines whether an element in b[ ] exists in a[ ]. Returns true if both conditions are met and false if either condition is not met.

2.1.1 Main menu flow chart


Start
cin>>lenA,cin>>lenB
cin>>a[ ],cin>>b[ ]
int i=0,temp,k=0
i<lenA?
YES
NO
temp=a[i]
i++
k<lenB?
YES
temp=b[k]?
YES
NO
k++
return true
return false
end
j<lenB
YES
temp=b[j]
j++
k<lenA?
temp=a[k]?
YES
NO
k++
return true
return false
end

2.1.2 On an input

In the main interface, the user needs to input the length of the two arrays and the contents of the array. According to the requirements, the length of the two arrays should be different. Therefore, we need to input:

1) int a [ ]
2) int lenA
3) int b [ ]
4) int lenB

Where, the array cannot be of any length other than a number

2.1.3 On Outputs

The program will print out the "Same" if the function returns true; The program will print out the "same" if the function returns true; print out "different" if the function returns false.

2.1.4 Variables

In both for loops, we use “temp” to replace the elements in each array, “i” is used to store each element, and “k” is used to determine whether the elements in both arrays are the same.

2.1.5 Data structure

We define two arrays and leave the choice of their length to the user. The user can enter two arrays of different lengths. Next, we will make a judgment based on the arrays entered by the user.

2.1.6 Algorithm

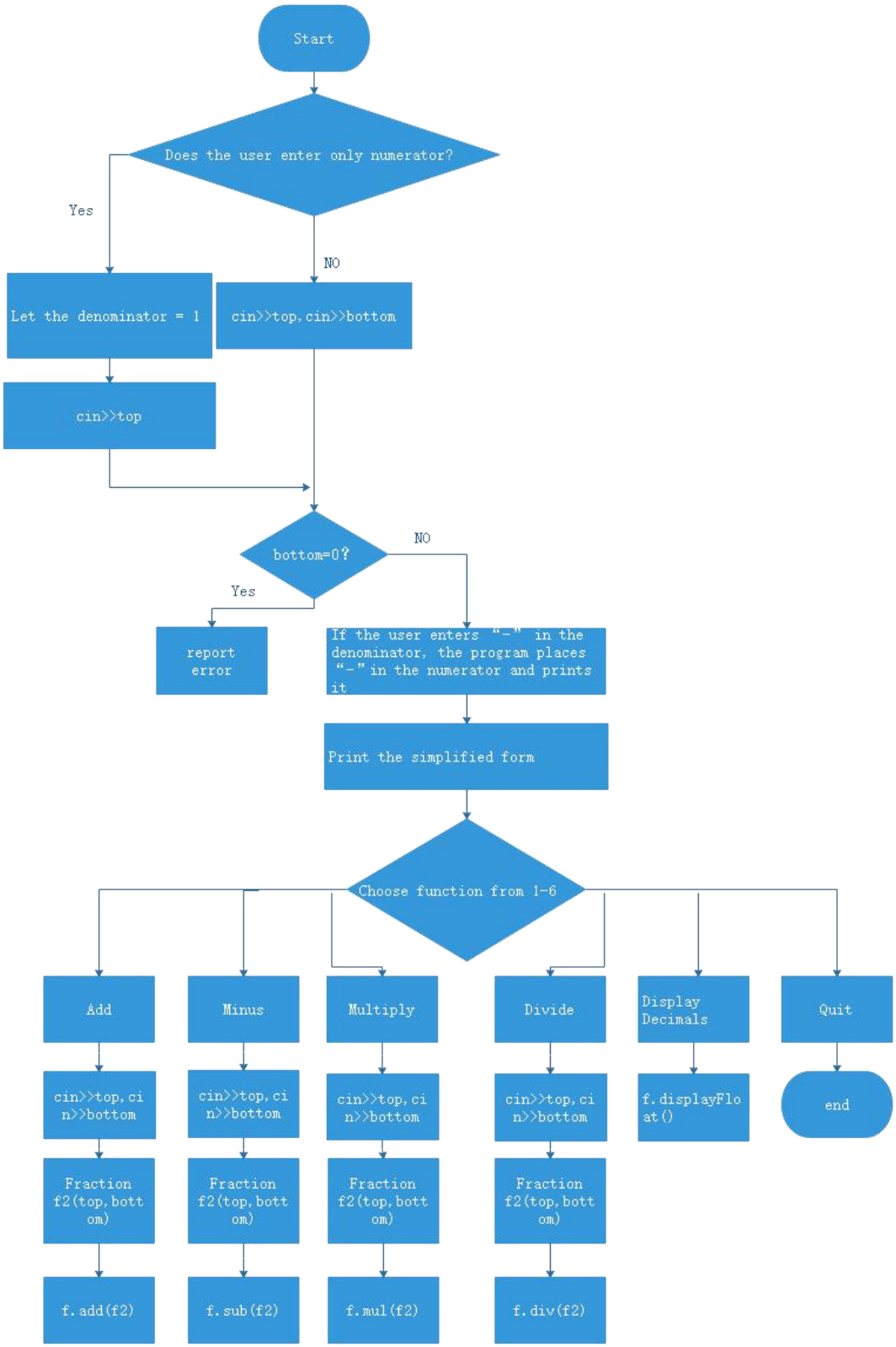
Here is part of my procedure(without ;) for determining whether two arrays are the same. I choose to replace the elements in each array with “temp”

```
for(int i=0;i<lenA;i++){
        int flag=0
        int temp=a[i]
        for(int k=0;k<lenB;k++){
            if(temp==b[k]){
                flag=1
                break
            }
        }
        if(flag==0){
            return false;
        }
    }
```

2.2 For exercise2

We first let the user chooses whether to enter nothing or the numerator and denominator or just enter the numerator (The denominator defaults to 1) which defaults to 0/1. Next, we're going to do what the user chooses(add, subtract, multiply and divide), and every time the user enters a fraction, we're going to make sure that the **denominator is not zero**.

2.2.1 Main flow chart


Start
Does the user enter only numerator?
Yes
NO
Let the denominator = 1
cin>>top,cin>>bottom
cin>>top
bottom=0?
NO
Yes
report error
If the user enters "-" in the denominator, the program places "-" in the numerator and prints it
Print the simplified form
Choose function from 1-6
Add
Minus
Multiply
Divide
Display Decimals
Quit
cin>>top,cin>>bottom
cin>>top,cin>>bottom
cin>>top,cin>>bottom
cin>>top,cin>>bottom
f.displayFloat()
end
Fraction f2(top,bottom)
Fraction f2(top,bottom)
Fraction f2(top,bottom)
Fraction f2(top,bottom)
f.add(f2)
f.sub(f2)
f.mul(f2)
f.div(f2)

2.2.2 On an input

In the beginning, we would like to let the user choose the form of the input fraction.For example：

1) Fraction a; represents 0/1

2) Fraction b(3,4); represents 3/4

3) Fraction c(5); represents 5/1

After successful input, we need to select the function which user chosen. If we select these functions, we need to re-enter a fraction to operate on both fractions. The main functions include:

1) Add the two fractions

2) Subtract the two fractions

3) Multiply the two fractions

4) Divide the two fractions

5) Change the fraction to a decimal

6) Exit the interface

When the user selects the above function, the program will execute according to the function. It is important to note that every time the user enters a fraction, the program must detect the denominator of the user's input, where the denominator cannot be 0, and if the user enters a 0, the program must be able to prompt the user for input error.

2.2.3 On an output

When the user input fraction which can be simplified, for example: 3/9. The program will convert it to 1/3, and print the result into the interface.

When the user enters a denominator that contains a "-" , the program puts the minus sign on the numerator and prints it out.

After the user chooses the function, the program needs to execute the operation according to the user's choice. After each operation, the user has the option to convert the fraction to decimal.

When the user chooses to quit, the system will automatically quit.

2.2.4 Variables

We declare “top” and “bottom” in our private class to represent the numerator and denominator of a fraction.

In the common divisor function, we define two variables a and b. In the divisor function, we find the greatest common divisor.

In the add function, we newly define b and t to store common divisors and calculated results, and finally get the final result through common divisors.

2.2.5 Data structure

We used class Fraction and set top and bottom as our private classes. For each of these fractions, we're going to simply. If the user enters a "-" in the denominator, we'll put the minus sign in the numerator. If the user chooses addition, subtraction, multiplication or division, we will first calculate the greatest common divisor of the two fractions and divide. When we would like to get the greatest common divisor, we define two new variables, a and b. We also need class Fraction to add, subtract, multiply and divide

2.2.6 Algorithm

1) Part of the main functions for adding two fractions:

```
void Fraction::add(Fraction a){
        int b
        int t
        b = gcd(bottom, a.bottom)
        b = (bottom * a.bottom) / b
        t = (top)*(b/bottom)+(a.top)*(b/a.bottom);
        lowest(b,t);//reduction of a fraction
        bottom=b
        top=t
        display()
    }
```

2)Part of the main functions for the **reduction of a fraction**:

```
void lowest(int& den3, int& num3){
    // For example 2/4: common_factor=2;
     int common_factor = gcd(num3, den3);
     den3 = den3 / common_factor;//The numerator divided by 2
     num3 = num3 / common_factor;//The denominator divided by 2
}
```

3.Design

3.1 For exercise1

1. We first declare two arrays a[ ] and b[ ] and ask the user to enter their length. At the same time, we check whether the user input is correct.

2. We change each element of a[ ] to “temp” one by one, and compare it with the elements in b[ ]. Returns false if the element represented by “temp” cannot be found in b[ ]

3. We change each element of b[ ] to “temp” one by one, and compare it with the elements in a[ ]. Returns false if the element represented by “temp” cannot be found in a[ ]

3. If both conditions are satisfied, the program prints “same”.If one condition is not satisfied, print different

3.2 For exercise 2

1) We first declare the numerator and denominator in the private class: top and bottom

Next, we ask the user for input methods, including:

1) Fraction a; represents 0/1

2) Fraction b(3,4); represents 3/4

3) Fraction c(5); represents 5/1

2) In the public class, we perform the operation of placing the minus sign on the numerator. This part of the operation involves the overloading. Function overloading is a special case in which C++ allows several similar functions with the same name to be declared in the same scope. These functions must have different parameter lists (number of arguments, type, order). This is often used to handle the problem of implementing similar functions with different data types.

3) After placing the minus sign in the numerator, we declare several functions, such as addition, subtraction, multiplication and division, that we need to use later.

4) We set up a function for the greatest common divisor

5) Through the function established above, we establish the divisor function, and calculate on the basis of obtaining the common factor

6)Next, we need to create the addition function, and once we get the result, we also need to simply the result

7) Subtraction is achieved by referring to the addition function by adding a minus sign to the numerator

8) After the user enters the second fraction, we multiply the second fraction by the first fraction

9) For division, after the user enters the fraction, we switch the numerator and denominator of the fraction, referencing the multiplication function

10) In building the main function, we first need to ask the user to input the numerator and denominator. If we want to do something later, we need to ask the user to enter it a second time. Based on this, we set up a **switch** function to select the function

4.Testing

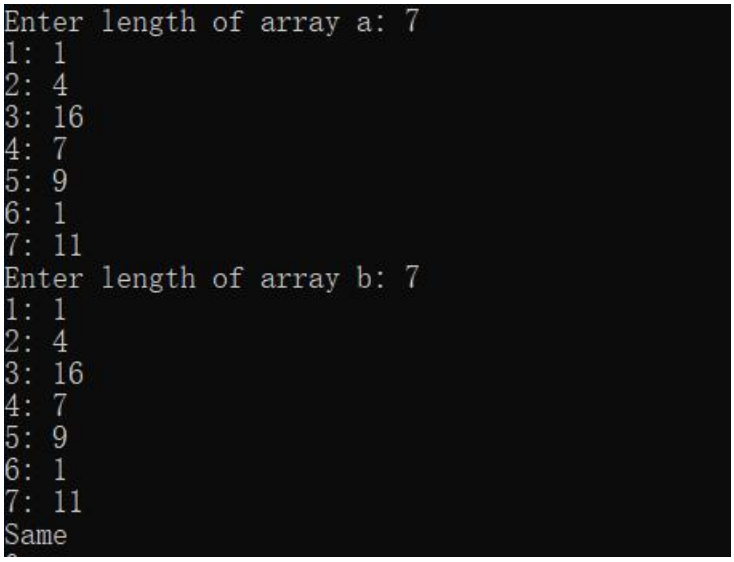
4.1 For exercise 1

4.1.1 Major functional tests

Although the number of elements in the two arrays should be different, my program can recognize the same or different length of arrays.

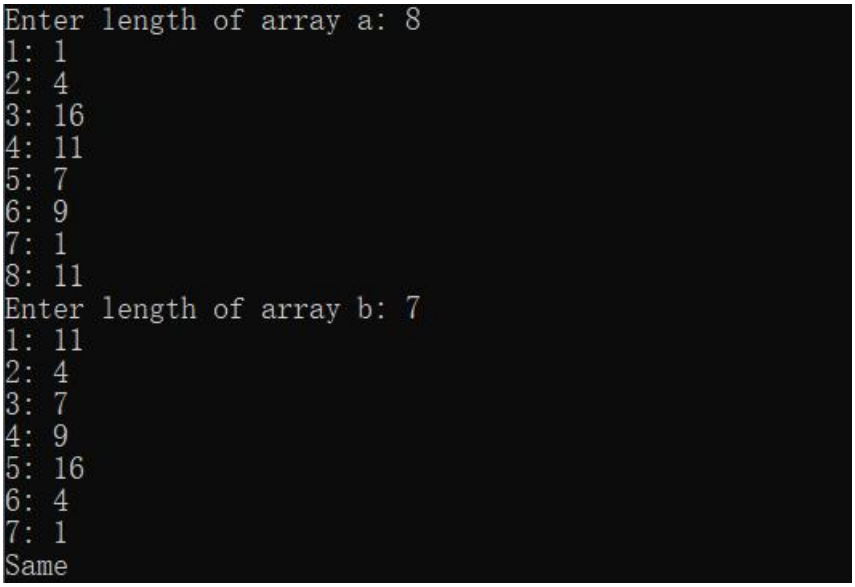
Test 1

We first input two same arrays, and found that after the program judgment, the program output is True. For example, we input two arrays {1, 4, 16, 7, 9, 1, 11} and {1, 4, 16, 7, 9, 1, 11} and the program will output “Same”



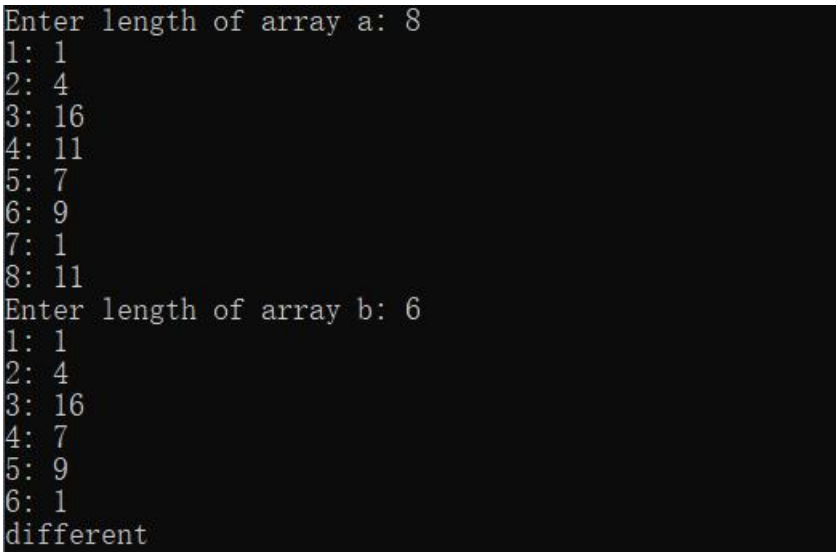
Test 2

We input two same arrays(Elements are in different order and some elements are duplicated), and found that after the program judgment, the program output is True. For example, we input two arrays {1, 4, 16, 11, 7, 9, 1, 11} and {11, 4, 7, 9, 16, 4, 1} and the program will output “Same”



Test 3

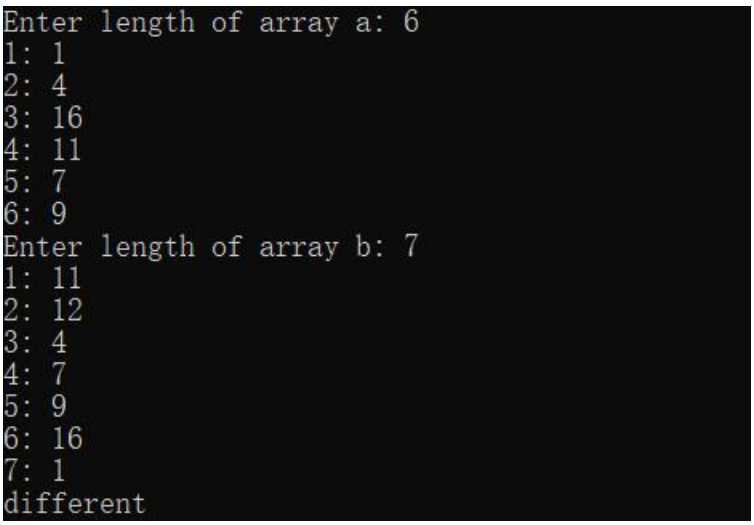
We first input two different arrays(The second array is missing elements), and found that after the program judgment, the program output is Different. For example, we input two arrays {1, 4, 16, 11, 7, 9, 1, 11} and {1, 4, 16, 7, 9, 1} and the program will output “Different”



Test 4

We input two different arrays(The second array has more elements than the first), and found that after the program judgment, the program output is

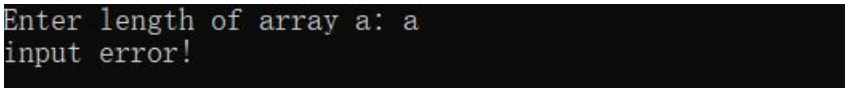
Different. For example, we input two arrays {1, 4, 16, 11, 7, 9} and {11,12, 4, 7, 9, 16, 1} and the program will output “Different”



4.1.2 Report error section (Users input error message)

Test 5

When inputting the length of arrays, we input the wrong character which is non-number. After the program’s judgement, it reports an error. For example, when you input lenA, you type a. The program outputs “input error!”.

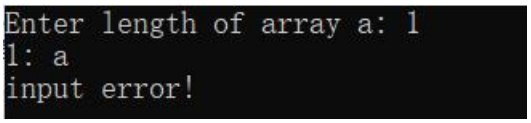


Test 6

When inputting the length of arrays, we input the wrong character which is a special character. After the program’ s judgement, it reports an error. For example, when you input lenA, you type @. The program outputs “input error!”.

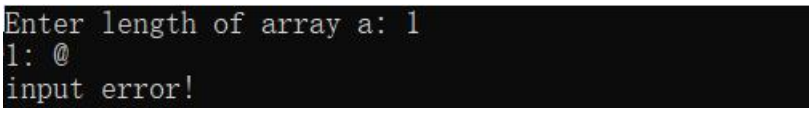
Test 7

When inputting the elements of arrays, we input the wrong character which is a non-number. After the program’s judgement, it reports an error. For example, when you input a[ ], you type a. The program outputs “input error!”.



Test 8

When inputting the elements of arrays, we input the wrong character which is a special character. After the program’s judgement, it reports an error. For example, when you input a[ ], you type @. The program outputs “input error!”.

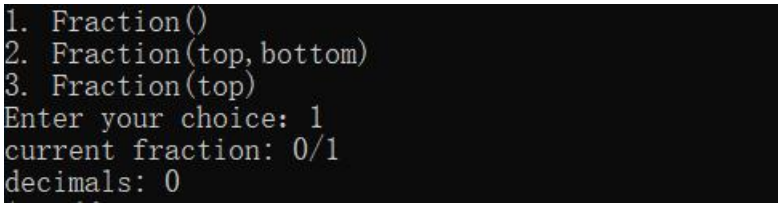


4.2 For exercise 2

4.2.1 Fundamental requirements

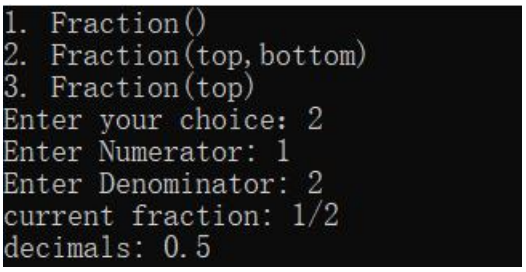
Test 1

When the user initially chooses to enter nothing, the program automatically assumes that the numerator of the user's input is 0 and the denominator is 1. For example, if the user does not input, the program outputs 0/1 and the decimals is 0.



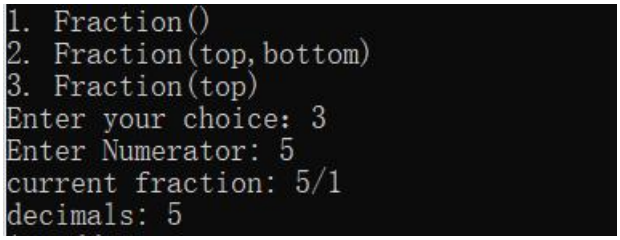
Test 2

The user inputs top and bottom when initially selecting the input method, and the program automatically changes the user's input numbers into the numerator and denominator. For example, the input numerator is 1, the input denominator is 2. The output of the program is 1/2 and the decimals is 0.5.



Test 3

When the user initially chooses the input method, he chooses Fraction(Top). He inputs only one number and the program automatically changes the number entered by the user into a numerator. For example, if you input 5, the program outputs 5/1 and the decimals is 5.



Test 4

After entering a fraction, the user selects the addition operation. And again, the user enters a fraction. For example, the user inputs 1/2 the first time, 1/3 the second time, and the program outputs 5/6.

```
1. Fraction()
2. Fraction(top,bottom)
3. Fraction(top)
Enter your choice: 2
Enter Numerator: 1
Enter Denominator: 2
current fraction: 1/2
decimals: 0.5
1. Add
2. Subtract
3. Multiple
4. Divide
5. Display Decimals
0. Quit
Your Choice: 1
1. Fraction()
2. Fraction(top,bottom)
3. Fraction(top)
Enter your choice: 2
Enter Numerator: 1
Enter Denominator: 3
current fraction: 5/6
```

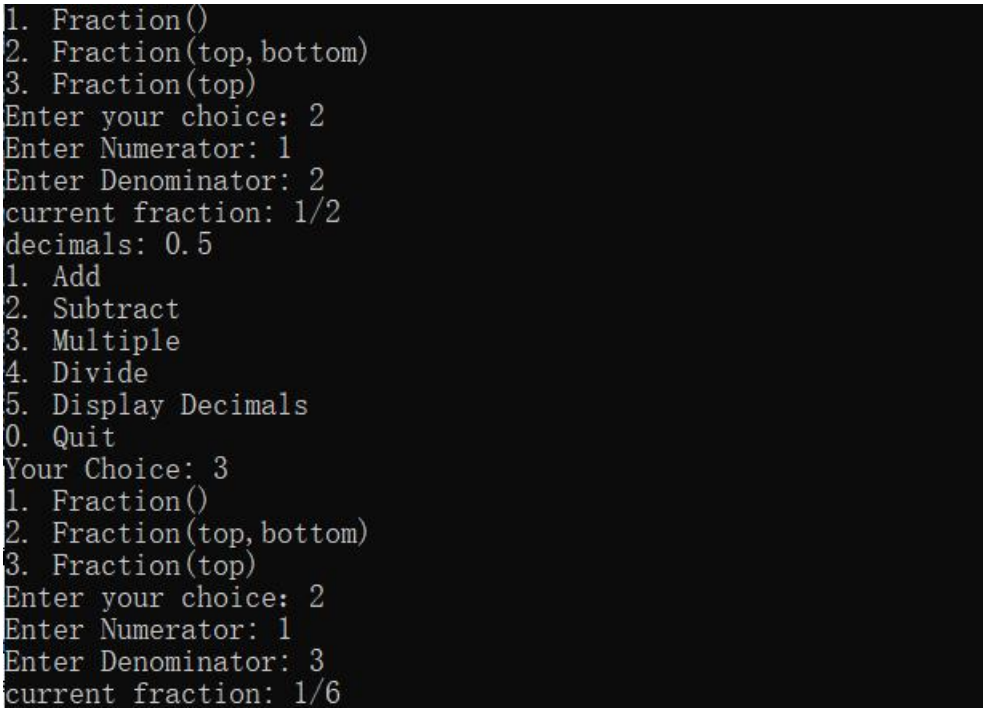
Test 5

After entering a fraction, the user selects the subtraction operation. And again, users enter a fraction. For example, if the user inputs 1/2 on the first time, 1/3 on the second time, and the program outputs 1/6. Since the result is a positive number, we can determine that the first fraction is larger.

```
1. Fraction()
2. Fraction(top,bottom)
3. Fraction(top)
Enter your choice: 2
Enter Numerator: 1
Enter Denominator: 2
current fraction: 1/2
decimals: 0.5
1. Add
2. Subtract
3. Multiple
4. Divide
5. Display Decimals
0. Quit
Your Choice: 2
1. Fraction()
2. Fraction(top,bottom)
3. Fraction(top)
Enter your choice: 2
Enter Numerator: 1
Enter Denominator: 3
current fraction: 1/6
```

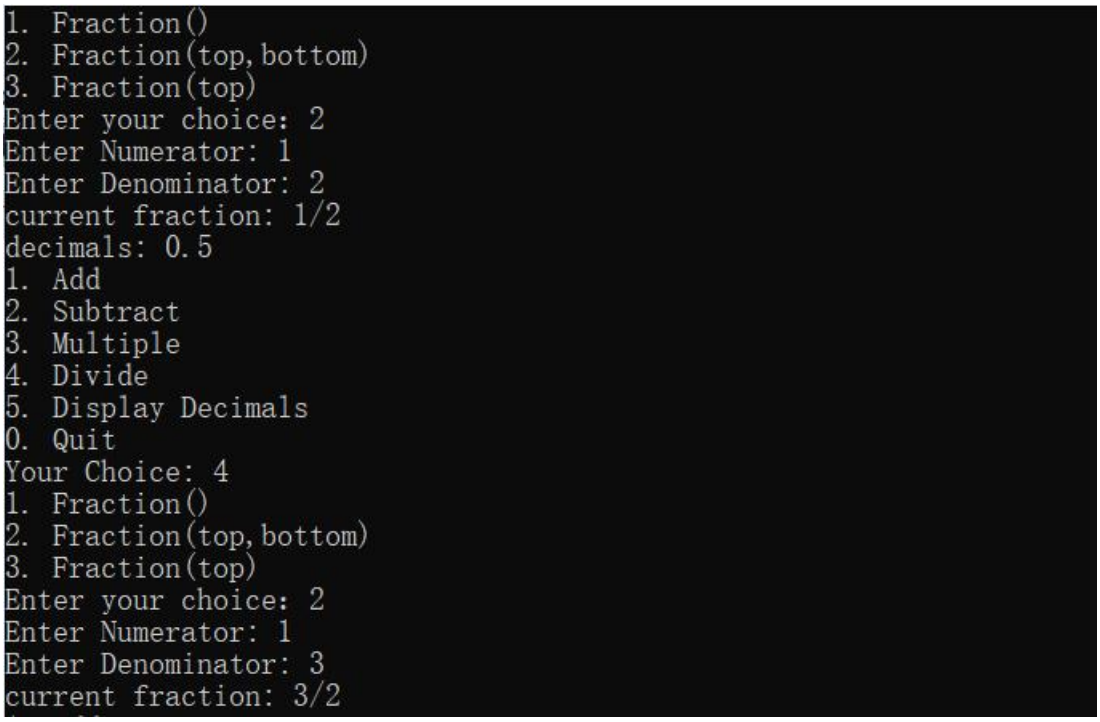
Test 6

After entering a fraction, the user selects the multiplication operation. And again, the users enter a fraction. For example, the user inputs 1/2 the first time, 1/3 the second time, and the program outputs 1/6.



Test 7

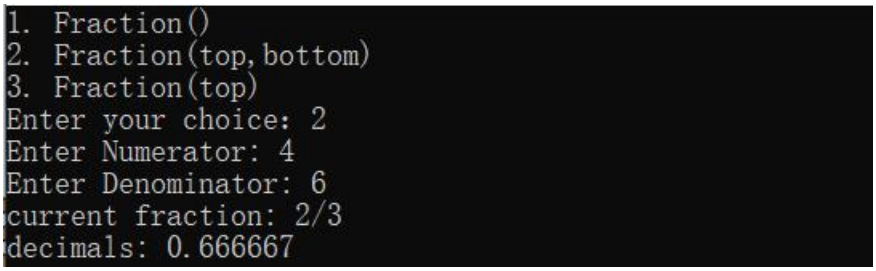
After entering a fraction, the user selects the division operation. And again, users enter a fraction. For example, the user inputs 1/2 the first time, 1/3 the second time, and the program outputs 3/2



4.2.2 Advanced requirements

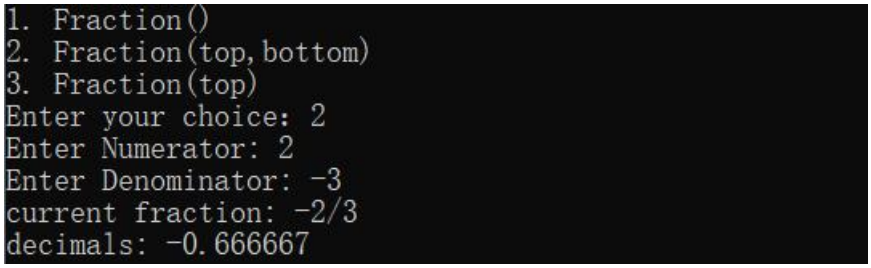
Test 8

Simplify the input. If the fractions entered by the user can be simplified, the program will show the user the result of the simplification. For example, the user inputs 4/6, and the program outputs 2/3



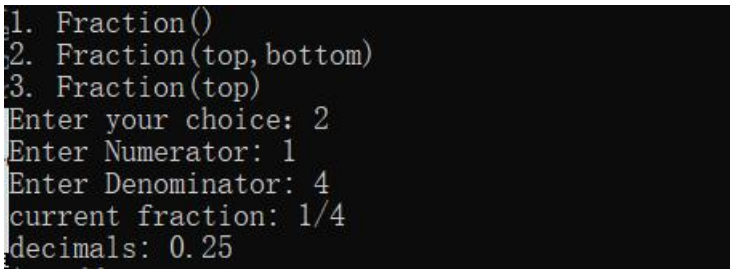
Test 9

Put the minus sign in the numerator. If the user enters a “-” sign in the denominator, we need to put the "-" in the numerator and print it. For example, the user inputs 2/-3, and the program outputs -2/3



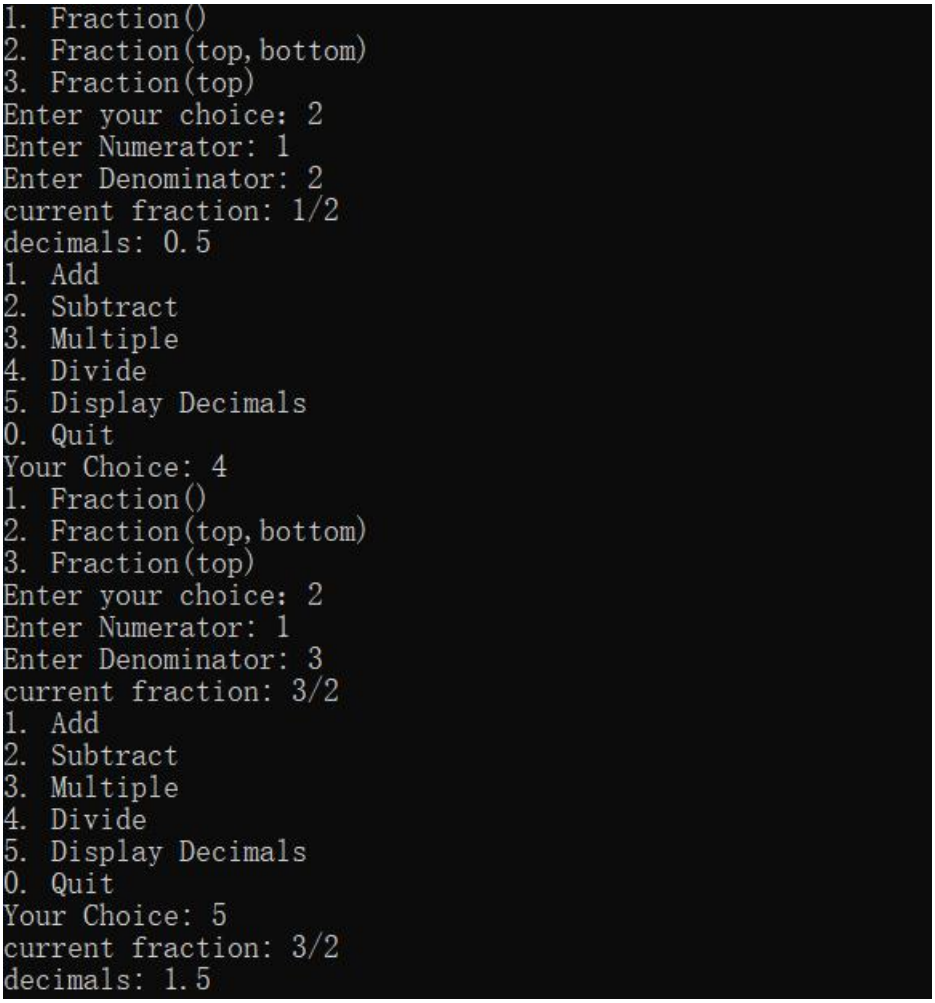
Test 10

We came up with methods to convert between decimals and fractions. For example, we input 1/4 and the program outputs 0.25.



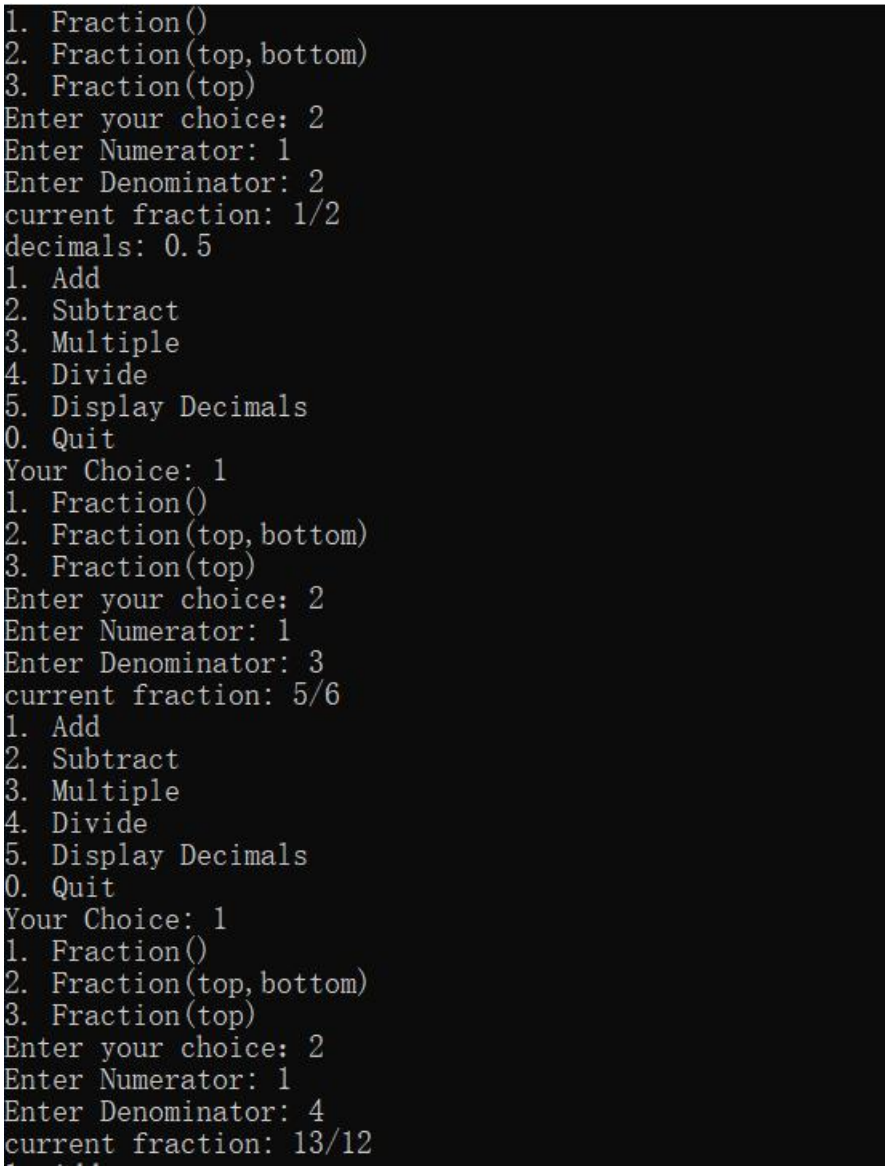
Test 11

After operation for addition, subtraction, multiplication, and division , we can also convert between decimals and fractions.For example, we choose the “add” and the user inputs 1/2 the first time, 1/3 the second time, the program outputs 3/2 and the decimals will be 1.5.



Test 12

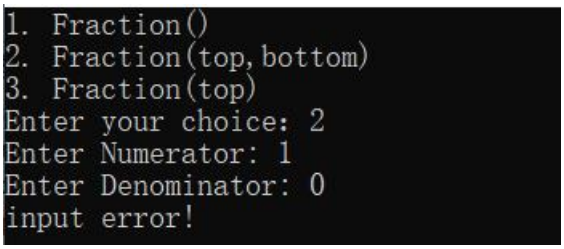
We can add, subtract, multiply and divide many times. For example, we choose the addition operation. On the first time, we enter 1/2, on the second time, we enter 1/3, and on the third time, we enter 1/4. After the calculation, we get 13/12.



4.2.3 Report error section (Users input error message)

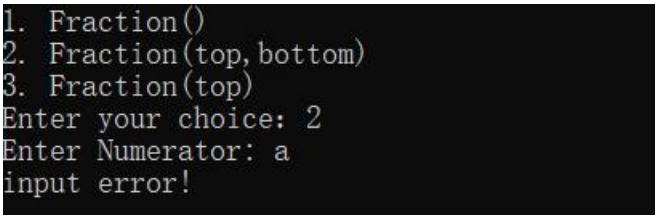
Test 13

The user may have entered a 0 in the denominator. The system tells the user that the error was made and asks the user to re-enter it.



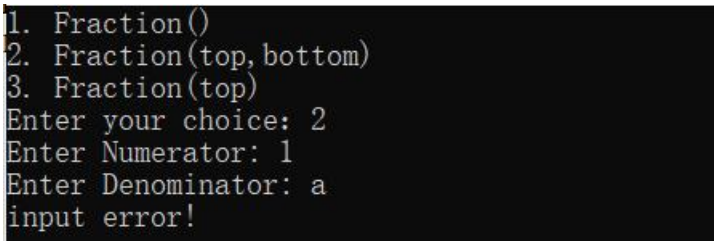
Test 14

Since the user may have typed incorrectly, the program needs to recognize the error and ask the user to retype. For example, when the user typed the numerator, he typed a. The program thinks the user made an error and asks the user to retype.



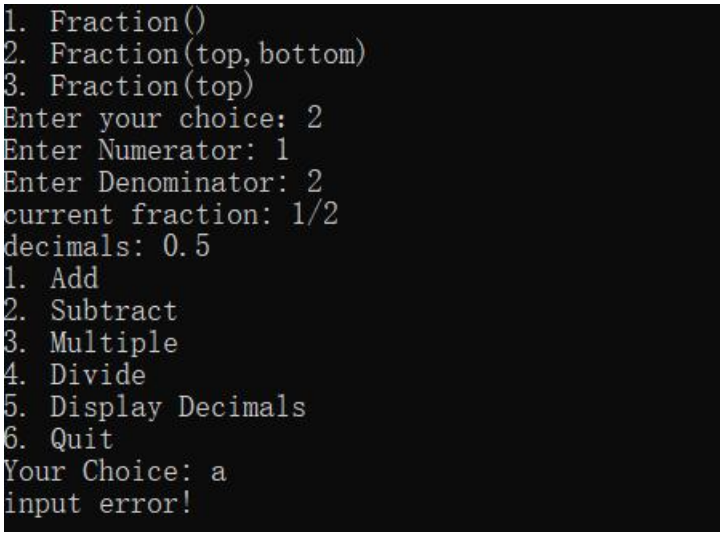
Test 15

Since the user may have typed incorrectly, the program needs to recognize the error and ask the user to retype. For example, when the user typed the denominator, he typed a. The program thinks the user made an error and asks the user to retype.



Test 16

Since the user may have typed incorrectly, the program needs to recognize the error and ask the user to retype. For example, when selecting a function, the user typed “a” when selecting among the six options. The program thinks the user made a typo and asks the user to retype.



Test 17

Since the user may have typed incorrectly, the program needs to recognize the error and ask the user to retype. For example, when selecting a function, the user typed “7” when selecting among the six options. The program thinks the user made a typo and asks the user to retype.