Assignment 8.1 [75 points]

Note: No documentation is required on this assignment

Write a `fraction` class whose objects will represent fractions. For this assignment you aren't required to reduce your fractions. You should provide the following member functions:

1. A set() operation that takes two integer arguments, a numerator and a denominator, and sets the calling object accordingly.
2. Arithmetic operations that add, subtract, multiply, and divide `fraction`s. These should be implemented as value returning functions that return a `fraction` object. They should be named `addedTo`, `subtract`, `multipliedBy`, and `dividedBy`. In these functions you will need to declare a local "fraction" variable, assign to it the result of the mathematical operation, and then return it.
3. A boolean operation named `isEqualTo` that compares two `fraction` objects for equality. Since you aren't reducing your fractions, you'll need to do this by cross-multiplying. A little review: if numerator1 * denominator2 equals denominator1 * numerator2, then the fractions are equal.
4. An output operation named `print` that displays the value of a `fraction` object on the screen in the form numerator/denominator.

Your class should have exactly two data members, one to represent the numerator of the fraction being represented, and one to represent the denominator of the fraction being represented.

Here's a hint for how you will set up your arithmetic operation functions: You need two fractions. One is the parameter, one is the calling object. The function multiplies the calling object times the parameter and returns the result. In some ways it is similar to the comesBefore() function from the lesson. That function also needs two fractions, and one is the calling object and one is the parameter.

When adding or subtracting `fraction`s, remember that you must first find the common denominator. The easy way to do this is to multiply the denominators together and use that product as the common denominator.

I am providing a client program for you below. You should copy and paste this and use it as your client program. The output that should be produced when the provided client program is run with your class is also given below, so that you can check your results.

I strongly suggest that you design your class incrementally. For example, you should first implement only the set function and the output function, and then test what you have so far. Once this code has been thoroughly debugged, you should add additional member functions, testing each one thoroughly as it is added. You might do this by creating your own client program to test the code at each stage; however, it would probably be better to use the provided client program and comment out code that relates to member functions that you have not yet implemented.

As you can see from the sample output given below, you are not required to reduce fractions or change improper fractions into mixed numbers for printing. Just print it as an improper fraction. You are also not required to deal with negative numbers, either in the numerator or the denominator.

Here is the client program.

Here is the client program.

```
#include <iostream>
using namespace std;

int main()
{
    fraction f1;
    fraction f2;
    fraction result;

    f1.set(9, 8);
    f2.set(2, 3);

    cout << "The product of ";
    f1.print();
    cout << " and ";
    f2.print();
    cout << " is ";
    result = f1.multipliedBy(f2);
    result.print();
    cout << endl;

    cout << "The quotient of ";
    f1.print();
    cout << " and ";
    f2.print();
    cout << " is ";
    result = f1.dividedBy(f2);
    result.print();
    cout << endl;

    cout << "The sum of ";
    f1.print();
    cout << " and ";
    f2.print();
    cout << " is ";
    result = f1.addedTo(f2);
    result.print();
    cout << endl;

    cout << "The difference of ";
    f1.print();
    cout << " and ";
    f2.print();
    cout << " is ";
    result = f1.subtract(f2);
    result.print();
    cout << endl;

    if (f1.isEqualTo(f2)){
        cout << "The two fractions are equal." << endl;
    } else {
        cout << "The two fractions are not equal." << endl;
    }
}
```

This client should produce the output shown here:

```
The product of 9/8 and 2/3 is 18/24
The quotient of 9/8 and 2/3 is 27/16
The sum of 9/8 and 2/3 is 43/24
The difference of 9/8 and 2/3 is 11/24
The two fractions are not equal.
```

You may not change the client program in any way. Changing the client program will result in a grade of 0 on the project.

Note: in future weeks you will be submitting the client program and class in separate files, but for this week you will be putting all of your code, class and client program, into a single file to submit it. The class declaration will come first, followed by the definitions of the class member functions, followed by the client program.

Submit Your Work

Name your source code file(s) according to the assignment number (a1_1.cpp, a4_2.cpp, etc.). Execute each program and copy/paste the output into the bottom of the corresponding source code file, making it into a comment. Use the Assignment Submission link to submit the source file(s). When you submit your assignment there will be a text field in which you can add a note to me (called a "comment", but don't confuse it with a C++ comment). In this "comments" section of the submission page let me know whether the program(s) work as required.

Assignment 9.1 [75 points]

For this assignment you will be building on the fraction class you began last week. You'll be making four major changes to the class.

1. **[15 points]** Delete your set() function. Add two constructors, a default constructor that assigns the value 0 to the `fraction`, and a constructor that takes two parameters. The first parameter will represent the initial numerator of the `fraction`, and the second parameter will represent the initial denominator of the `fraction`.

   Since fractions cannot have denominators of 0, the default constructor should assign 0 to the numerator and 1 to the denominator.

2. **[15 points]** Add the `const` keyword to your class wherever appropriate. Your class may still work correctly even if you don't do this correctly, so this will require extra care!!

3. **[5 points]** Add a private "simplify()" function to your class and call it from the appropriate member functions. (There will probably be 5 places where you need to call it.) The best way to do this is to make the function a void function with no parameters that reduces the calling object.

   As you can see from the sample output given below, you are still not required to change improper fractions into mixed numbers for printing. Just print it as an improper fraction. Make sure that your class will reduce ANY fraction, not just the fractions that are tested in the provided client program. Fractions should not be simply reduced upon output, they should be stored in reduced form at all times. In other words, you should ensure that all fraction objects are reduced before the end of any member function. You are also not required to deal with negative numbers, either in the numerator or the denominator.

   You must create your own algorithm for reducing fractions. Don't look up an already existing algorithm for reducing fractions or finding GCF. The point here is to have you practice solving the problem on your own. In particular, don't use Euclid's algorithm. Don't worry about being efficient. It's fine to have your function check every possible factor, even if it would be more efficient to just check prime numbers. Just create something of your own that works correctly on ANY fraction.

   Note: this part of the assignment is worth 5 points. If you are having trouble keeping up with the class, I suggest you skip this part and take the 5 point deduction.

4. **[20 points]** Put the client program in a separate file from the class, and divide the class into specification file (fraction.h) and implementation file (fraction.cpp), so your code will be in 3 separate files.

5. Add documentation to your assignment. Be sure to carefully read section 1D of the Style Conventions, "Commenting in Classes".

Your class should still have exactly two data members.

I am providing a client program for you below. You should copy and paste this and use it as your client program. The output that should be produced when the provided client program is run with your class is also given below, so that you can check your results. Since you are not writing the client program, you are not required to include comments in it.

Here is the client program.

```
#include <iostream>
#include "fraction.h"
using namespace std;
```

```
int main()
{
    fraction f1(9,8);
    fraction f2(2,3);
    fraction result;

    cout << "The result starts off at ";
    result.print();
    cout << endl;

    cout << "The product of ";
    f1.print();
    cout << " and ";
    f2.print();
    cout << " is ";
    result = f1.multipliedBy(f2);
    result.print();
    cout << endl;

    cout << "The quotient of ";
    f1.print();
    cout << " and ";
    f2.print();
    cout << " is ";
    result = f1.dividedBy(f2);
    result.print();
    cout << endl;

    cout << "The sum of ";
    f1.print();
    cout << " and ";
    f2.print();
    cout << " is ";
    result = f1.addedTo(f2);
    result.print();
    cout << endl;

    cout << "The difference of ";
    f1.print();
    cout << " and ";
    f2.print();
    cout << " is ";
    result = f1.subtract(f2);
    result.print();
    cout << endl;

    if (f1.isEqualTo(f2)){
        cout << "The two fractions are equal." << endl;
    } else {
        cout << "The two fractions are not equal." << endl;
    }
```

```
    const fraction f3(12, 8);
    const fraction f4(202, 303);
    result = f3.multipliedBy(f4);
    cout << "The product of ";
    f3.print();
    cout << " and ";
    f4.print();
    cout << " is ";
    result.print();
    cout << endl;
}
```

This client should produce the output shown here:

```
The result starts off at 0/1
The product of 9/8 and 2/3 is 3/4
The quotient of 9/8 and 2/3 is 27/16
The sum of 9/8 and 2/3 is 43/24
The difference of 9/8 and 2/3 is 11/24
The two fractions are not equal.
The product of 3/2 and 2/3 is 1/1
```

You may not change the client program in any way. Changing the client program will result in a grade of 0 on the project.

Submit Your Work

Name your source code file(s) fraction.cpp and fraction.h. Execute the given client program and copy/paste the output into the bottom of the implementation file, making it into a comment. Use the Assignment Submission link to submit the source file(s). When you submit your assignment there will be a text field in which you can add a note to me (called a "comment", but don't confuse it with a C++ comment). In this "comments" section of the submission page let me know whether the class works as required.

Assignment 10.1 [75 points]

Write a program that reads in non-negative integers and stores and displays distinct numbers (i.e., if a number appears multiple times, it is stored and displayed only once). Your program should allow up to 1000 distinct numbers to be stored and displayed. Use the following algorithm (this is required!): Read each number and store it in an array if it is new. If the number is already in the array, ignore it. The user will indicate that they are done entering numbers by entering a negative number. Here is a sample run:

```
Enter a non-negative integer (negative to quit): 1
Enter a non-negative integer (negative to quit): 2
Enter a non-negative integer (negative to quit): 3
Enter a non-negative integer (negative to quit): 2
Enter a non-negative integer (negative to quit): 1
Enter a non-negative integer (negative to quit): 6
Enter a non-negative integer (negative to quit): 3
Enter a non-negative integer (negative to quit): 4
Enter a non-negative integer (negative to quit): 5
Enter a non-negative integer (negative to quit): 2
Enter a non-negative integer (negative to quit): -4
You entered:
1 2 3 6 4 5
```

To get credit for this assignment you must use appropriate decomposition! You should have a function to read the numbers and a function to print the resulting array. The function that reads the numbers should call an additional function that returns a bool value indicating whether a number is already in the array. At the conclusion of the call to the function that reads the numbers, the array MUST contain only distinct numbers.

Additionally, **you must not sort the array**. The numbers must appear in the same order in which they were typed.

Submit Your Work

Name your source code file(s) according to the assignment number (a1_1.cpp, a4_2.cpp, etc.). Execute each program and copy/paste the output into the bottom of the corresponding source code file, making it into a comment. Use the Assignment Submission link to submit the source file(s). When you submit your assignment there will be a text field in which you can add a note to me (called a "comment", but don't confuse it with a C++ comment). In this "comments" section of the submission page let me know whether the program(s) work as required.