PROGRAMMING PROJECT#3(b): INFORMATION AND SUBMISSION FORM - The "FRACTION CLASS" program (35pts) - due Friday April 24, 2015

This assignment information sheet is to be used ONLY by Santa Rosa Junior College students currently enrolled in CS 10 Sections 4293, 4524 and 5393 (Spring 2015).

Prior to working on this programming project.....please read Gaddis textbook Chapter 7 - **Introduction to Classes and Objects** (review section 7.1-7.11 on pages 407 - 452 and section 7.13-7.14 on pages 467-484). It is recommended that you also read the weekly lesson notes on **C++ Classes** on the class schedule pages, review any chapter resources provided on the **class calendar** for weeks#12 - 14 as well as any additional ancillary material provided by your instructor (**see downloads and resources page**) on the current topic under discussion for the week.

Originality of Work : Projects and assignments that show evidence of lack of originality of work in any of the following forms will receive a grade of 0.
Collaboration: Although discussing assignments and projects with fellow students is encouraged, these discussions should not be about specific code and one student should never look at another student's actual code. If significant portions of your code are similar to another student's code, both parties will receive a grade of 0 on the assignment. This is not an hypothetical situation. It happens every semester. If you find yourself hopelessly stuck, get help from me. In summary, there are three things that you should never do:

1. look at another student's code,
2. let another student look at your code, or
3. discuss code with another student in enough detail that parts of your program look identical.

Previous Semester Coursework: You should also never look at a solution to a programming exercise or a project from a previous semester, whether it is a sample solution or a student submission.
Other Sources: Many of the programs that you do this semester are classic problems that have been solved by thousands of students, and some may be found in various textbooks. If you find a book (or website, or other source) that includes a solution or partial solution to one of our assignment, you may review the source code but do not simply copy/paste the code into your program. Instead recreate your own algorithm and source code prior to submitting the assignment or project.

Instructions to all sections: Use your compiler to set-up a project and write the necessary multi-file C++ programs following the program specifications detailed below. After successfully compiling and debugging (if needed) your program, paste your source code and contents of your program output in the text input areas provided for the programming segments of this assignment. Complete all parts of this programming assignment and check for errors, prior to clicking submit.

Please make sure you have correctly input your name, section#, email address, SID (last four digits of your srjc student identification number) and the correct web form code visible to you. Upon submission you should receive a confirmation message. If you need further clarification please DO NOT hesitate to contact me, click to **email me** .

STUDENT NAME: **CS10 SECTION#:**

EMAIL ADDRESS: **SID#:** (last four digits)

Project#3(b)-Designing and implementing a C++ class data type: Fraction Class (35 points)..a multifile project

Write a `fraction` class whose objects will represent fractions. You should provide the following member functions:

1. **Two constructors**, a default constructor which 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`. (4 pts for specification and implementation)

2. **Arithmetic operations** that add, subtract, multiply, and divide fractions. These should be implemented as **four value returning functions** that return a `fraction` object. They should be named `AddedTo`, `Subtract`, `MultipliedBy`, and `DividedBy`. (16 pts for specification and implementation)
3. A **boolean operation** named `isGreaterThan` that compares two `fraction` objects to determine whether one `fraction` object is greater than the other . (4 pts for specification and implementation)
4. Another **boolean operation** named `isEqualTo` that compares two `fraction` objects for equality. (4 pts for specification and implementation)
5. An **output operation** named `print` that displays the value of a `fraction` object on the screen in the form numerator/denominator. (7 pts for specification and implementation)

NOTE: You may not implement the above member functions as inline functions but should include the function member declarations (or prototypes) in the class specification file (fraction.h) and then write the member function definitions in the implementation file (fraction.cpp).

Your class specification 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.

When adding or subtracting fractions, 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 into a file and use it as your client program that will be a part of your mutlitfile project. The output that should be produced when the provided client program is run with your class specification and implementation files is also given below, so that you can check your results. Since this project requires multiple files you will need to first create a new project in your compiler and then write the source code for your class specification file, the implementation file and then add the client code below to the project. For an example with explanations of such C++ class related multifile projects review the Software Engineering Tip: Seperating Class, Specification, Implementation, and Client Code in Chapter 7.11 in your Gaddis textbook (pgs. 446-452)

I strongly suggest that you design your class incrementally. For example, you should first implement only the constructors and the output function, comment out parts of the client code that is not needed 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 change improper fractions into mixed numbers for printing. Just print it as an improper fraction. You are, however, required to reduce fractions, as illustrated in the sample output. Fractions should be displayed in reduced form upon output. You are also not required to deal with negative numbers, either in the numerator or the denominator.

Here is the client program.

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

int main()
{
    fraction f1(9,8); //calling a parameterized class constructor
    fraction f2(2,3); //calling a parameterized class constructor
    fraction result;  //calling a default class constructor
    fraction f3; //calling a default class constructor

    cout << "The result starts off at ";
    result.print(); //calling an observer function
    cout << endl;

    cout << "The product of ";
    f1.print();
```

```
        cout << " and ";
        f2.print();
        cout << " is ";
        result = f1.MultipliedBy(f2); //a class binary operation - function
        result.print();
        cout << endl;

        f3 = result; //assignment

         if (f2.isGreaterThan(f3)){ //a class relational expression - boolean operation/function
            f2.print();
            cout <<" is greater than ";
            f3.print();
            cout<<endl;
        } else {
            f2.print();
            cout <<" is less than ";
            f3.print();
            cout<<endl;
        }

        cout << "The sum of ";
        f1.print();
        cout << " and ";
        f2.print();
        cout << " is ";
        result = f1.AddedTo(f2); //a class binary operation - function
        result.print();
        cout << endl;

        cout << "The difference of ";
        f1.print();
        cout << " and ";
        f2.print();
        cout << " is ";
        result = f1.Subtract(f2); //a class binary operation - function
        result.print();
        cout << endl;

        if (f1.isEqualTo(f2)){ //a class relational expression - boolean operation/function
            cout << "The two fractions are equal." << endl;
        } else {
            cout << "The two fractions are not equal." << endl;
        }

        const fraction f4(12, 8);
        const fraction f5(202, 303);

        result = f4.DividedBy(f5); //a class binary operation - function
        cout << "The quotient of ";
        f4.print();
        cout << " and ";
        f5.print();
        cout << " is ";
        result.print();
        cout << endl;

        system ("PAUSE");//exclude statement if not using Dev-C++
```

```
    return 0;
}
```

Compiling/running the client program above along with the fraction class specification and implementation file should produce the output shown below:

```
The result starts off at 0/1
The product of 9/8 and 2/3 is 3/4
2/3 is less than 3/4
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 quotient of 3/2 and 2/3 is 9/4
Press any key to continue . . .
```

After successfully completing this mutifile project (ie., your program output should match the above), use the three text boxes below to paste in your source code for the "fraction class" **specification file**, followed by source code for your "fraction class" **implementation file**, followed by the **output** produced when you ran your mutlitfile project using the client program above. **You may not change the client program in any way. Changing the client program will result in a grade of 0 on the project**.

*__important reminder:__ for multi-file projects to work in your C++ compiler/IDE environment ....make sure you create a new project folder and place all related files in the same folder. Also check that you have the appropriate #include statements in the class implementation and client code files. Xcode users should place these files in the debug folder. You may find a review of the instructor provided sample Abstract DataType project called "MoneyType" helpful when working on this - "Fraction class project".... follow this link => **Abstract_Data_Type.pdf** (see also class resources page for - a review of week#14 class lecture/lab recorded sessions along with with sample source code may be helpful).

Please also make sure to include appropriate function documentation as needed (see handout on **function documentation**) and separate your member functions defintions in the implementation file with at least 1 inches of whitespace. You must also include the constant key words for all member functions that are "observer type" of functions.

project 3 folder should have the following files:
fraction.h (class specification file)
fraction.cpp (class implementation file)
client.cpp (client code using the fraction class)

Turn in your source code for fraction.h, fraction.cpp and the program output. Note you DO NOT need to paste the client code already provided in the text areas below.

Paste your "source code" for Project#3b - fraction class "specification" - fraction.h in the text area below:

Paste your "source code" for Project#3b - fraction class "implementation" - fraction.cpp in the text area below:

Paste your program output for Project#3b - fraction class program in the text area below:

General comment area: You may use the text area below to provide any comments on this multifile programming project or if you are unclear and have a question on some of the new C++ syntax or semantics related content covered in recent weeks.

Please also include any comments on something new and useful (C++ related) that you discovered while developing and implementing algorithms (source code) to complete this C++ class project. *THANK YOU!*
(Comments are for Instructor use only and are not used for grading purposes)

Enter web form code **Web Form Protection Code**
reload image

RE-TYPE EMAIL ADDRESS:

Submit Reset

Sujan Sarkar - CS Instructor, Santa Rosa Junior College
Updated Saturday April 04, 2015