AST21105 Object-Oriented Programming & Design

Lab 11 – Classes and Objects III

A. Submission Details

In this lab, you are required to submit **ONE** C++ program in two parts (C# and C++ part) to solve the given problem shown in the section "Rational Class". To make the program implementation easier, you are suggested to write your program using Visual Studio .NET instead of doing it directly using paper & pencil. After you have completed the implementation, submit your program files (i.e. Rational.h, Rational.cpp and all files in the created C# project for this lab in a zip) using the electronic "drop-box" in Canvas, **One week** after your lab section is conducted. For details, please refer to the following:

Tuesday sections	*by*	*19:00 on 12 April 2016 (Tuesday)*
Wednesday sections	*by*	*19:00 on 13 April 2016 (Wednesday)*

You are reminded to double-check your solution to verify everything is correct before submission. You are also required to put your name and student ID at the beginning of your source file.

Important: You are only able to submit your work once.

B. Objective

Use C# GUI to call user-defined type's member functions constructed using C++.

C. Rational Class

In this part, you are required to:

i) Create a New Project and give your project the name Lab11.

ii) In this lab, you are required to create a class **Rational** that has the following features.

A **Rational** object should be able to hold **TWO** values, which represent numerator and denominator. Also, it consists of a number of functions to manipulate the data.

The UML class diagram is given below:

Rational
- numerator : int - denominator : int
+ initialize(int theNumerator, int theDenominator) : void + getNumerator() : int + getDenominator() : int + setNumerator(int theNumerator) : void + setDenominator(int theDenominator) : void + add(const Rational& right) const : Rational* + subtract(const Rational& right) const : Rational* + multiply(const Rational& right) const : Rational* + divide(const Rational& right) const : Rational* + reduce() : void + getRationalString() const : string + getFloatString() const : string

where

- numerator: data member to store the numerator of a fraction
- denominator: data member to store the denominator of a fraction
- initialize(int theNumerator, int theDenominator): a member function, which takes two arguments representing the targeted values of fraction. It initializes numerator using theNumerator, and denominator using theDenominator. Also, it calls member function: reduce to reduce the fraction.

- getNumerator(): a CONSTANT member function, which returns the data member numerator.
- getDenominator(): a CONSTANT member function, which returns the data member denominator.
- setNumerator(int theNumerator): a member function, which sets data member numerator by theNumerator.
- setDenominator(int theDenominator): a member function, which sets data member denominator by theDenominator.
- add(const Rational& right) const: a CONSTANT member function adds two Rational numbers. The result of the addition should be stored **in reduced form** and returned a pointer to this resulting object. (Hint: using reduce() function)
- subtract(const Rational& right) const: a CONSTANT member function subtracts two Rational numbers. The result of the subtraction should be stored **in reduced form** and returned a pointer to this resulting object. (Hint: using reduce() function)
- multiply(const Rational& right) const: a CONSTANT member function multiplies two Rational numbers. The result of the multiplication should be stored **in reduced form** and returned a pointer to this resulting object. (Hint: using reduce() function)
- divide(const Rational& right) const: a CONSTANT member function divides two Rational numbers. The result of the division should be stored **in reduced form** and returned a pointer to this resulting object. (Hint: using reduce() function)
- reduce(): a member function reduces the fraction into its simplest form. This can be done by dividing the numerator and denominator by their Greatest Common Divisor (GCD).
- getRationalString() const: a CONSTANT member function returns a string representation of a Rational number.
- getFloatString() const: a CONSTANT member function returns a floating-point string representation of a Rational number.

iii) Amain function (in main.cpp) has been provided for you to test out your class.

At startup, it prompts the user to input two fractions. The following is the sample output.

```
Enter numerator 1: 5
Enter denominator 1: 4
Enter numerator 2: 3
Enter denominator 2: 1
```

After the input is done, the system should clear the screen and show a text-based menu:

```
--------------------
Operation:
--------------------
1.    Add
2.    Subtract
3.    Multiply
4.    Divide
5.    Exit
--------------------

Choice:
```

After reading the choice, the program should perform the operation according to the user input. Example output should look like this:

```
a + b = 17/4
a + b = 4.25
Press any key to continue ...
```

```
a – b = -7/4
a – b = -1.75
Press any key to continue ...
```

```
a * b = 15/4
a * b = 3.75
Press any key to continue ...
```

```
a / b = 5/12
a / b = 0.416667
Press any key to continue ...
```

Hints:

How to convert int to string in C++?
The following is an example demonstrating how to convert an int to string in C++.

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

int main()
{
    int i = 10;  int j = 20;
    stringstream s;
    s << i << " " << j;
    cout << s.str() << endl;
    system("pause");
    return 0;
 }
```

int_to_string_new.cpp

iv) Construct a Graphical User Interface (GUI) for the **Rational** application using Visual C#, which should resemble to what is provided in the given main program.

v) Convert your C++ project to CLR (Class Library) and link it to the Graphical User Interface to provide a fully functional program.

Marking Scheme:

Graded items	Weighting
1. Correctness of program (i.e. whether your code is implemented in a way according to the requirements as specified.)	60%
2. Indentation	30%
3. Documentation (with reasonable amount of comments embedded in the code to enhance the readability.)	10%
	100%

Program Submission Checklist

Before submitting your work, please check the following items to see you have done a decent job.

Items to be checked	☑ / ☒
1. Did I put my name and student ID at the beginning of all the source files?	☐
2. Did I put reasonable amount of comments to describe my program?	☐
3. Are they all in .cpp extension and named according to the specification?	☐
4. Have I checked that all the submitted code are compliable and run without any errors?	☐
5. Did I zip my source files using Winzip / zip provided by Microsoft Windows? Also, did I check the zip file and see if it could be opened? ***(Only applicable if the work has to be submitted in zip format.)***	☐
6. Did I submit my lab assignment to Canvas?	☐

-End-