University of Wollongong

School of Computing and Information Systems

CSCI251/851 **Advanced Programming** **Autumn 2018**

Exercise 4 **Demo Required during your Week 11 Lab Class** **3 marks**

(Also, submit your work using the submit utility on banshee by 11.59pm Friday 18 May)

The purpose of this assignment is to experiment with:

- Designing classes
- Static members and static functions
- Handling exceptions within a program

Problem:

You are to implement a class called *Rational* for performing arithmetic with fractions. It uses integer variables to represent a fractional number in the private data of the class – i.e. the numerator and the denominator. (e.g. the fraction 2/4 should be stored in a *Rational* object as 2 in the numerator and 4 in the denominator). Your code should go in files: *rational.h* and *rational.cpp*. A driver program is provided in *main.cpp*.

Step-1

Implement a default constructor, a standard constructor and a copy constructor. The default constructor should initialize the class to 1/1. Also, implement public member functions for doing the following:

- Addition of two Rational numbers.
- Subtraction of two Rational numbers.
- Multiplication of two Rational numbers.
- Division of two Rational numbers.
- Printing Rational numbers in the form a/b where a is the numerator and b is the denominator.

An incomplete class declaration is provided in rational.h. The member function definitions should go in rational.cpp. main.cpp is setup to test your step-1 functions.

Step-2

Implement a standalone (non-class) function:

```
void printRationalAsFloating(const Rational &r);
```

for printing the passed *Rational* number as a floating point number. Note: for this to work you will have to make this function a friend function to class Rational. Modify the main() function to test this function.

Step-3

Implement a private static class data member named *Count* for maintaining a count of the number of instances of class *Rational* that exist. This can be done by simply incrementing and decrementing *Count* in the constructors and destructor respectively. Also provide a static class member function for accessing the static Count data member. Modify your main() procedure to test this static function.

Step-4

Provide exception handling to prevent division by zero. Your program should throw a string exception to anywhere where divide by zero can happen. You should catch this exception in the main() following the tests. The exception handler should print an appropriate error message and exit(). Modify you main to test the exception handler.

Submit:

Submit your files using the submit facility on UNIX as shown below:

```
$ submit -u login -c CSCI251 –a ex4 main.cpp rational.h rational.cpp
```

where 'login' is your UNIX login ID
Note: CSCI851 should also submit to –c CSCI251.

You must also demonstrate your program in your week 11 lab class. Failure to demo on time, without being granted an extension, will result in a 1 mark deduction for each week late. Late submissions without granted extension will receive a deduction of 0.5 marks for each day late.