Advanced Programming

Homework Assignment 1

Object Oriented Programming

General guidelines:

a. Submission must be done according to the submission guidelines – a document published in the course website – read them carefully!
b. Maximize readability and ensure indentation.
c. Do exactly as required in the questions.
d. Add functions and helper methods as necessary to ensure readability.
e. Use relevant names.
f. Use comments to document your functions and complex code parts. **Add an output example at the end of your file!**
g. Individual work and submission – paired submission is not allowed.

Important note: Unless otherwise specified, ever homework has no more than one week for submission.
Open submission boxes past the deadline are not permission for late submission.

Question 1:

Define and implement a class called Rational that represents rational numbers.
Note: The denominator must not be 0 or negative.

The class must include the following fields:

- numerator
- denominator

The class must also include the following methods:

- Constructors:
  - An assignment constructor (ctor) with parameters that default to 0 for the numerator and 1 for the denominator. The implementation must contain the following output:
    print: in constructor
  - A copy constructor (copy-ctor). The implementation must contain the following output:
    print: in copy constructor
- For every field:
  - A setter.
    The denominator cannot be 0, so if the argument given is 0 set the denominator to 1. The denominator cannot be negative, so if the argument given is negative, reset the numerator accordingly: If it was positive make it negative and if it was negative make it positive.
  - A getter.
- A method print for printing the rational number. The printing is to be in the format: numerator/denominator (For example: 1/2, 3/4, 54/56 etc.) according to the original values and not the values after reduction.
- A boolean method called equal that compares two Rational numbers and tests if they are the same. Note: In this context the same means that the numerators are equal and the denominators are equal (1/2 and 3/6 are not considered equal).

- A method called makeEquals that receives a Rational object and modifies it to equal the calling object.
  Note: The method is of return type void. The argument is cbr so that there is no call to the copy-ctor.
- A **helper** method for reducing the fraction:
  - Function header: void reduce();
  - The function is to reduce the calling object. This means, for example, that given the following code:

    ```
    Rational rat(2,4);
    rat.reduce();
    ```

    The resulting numerator for rat will be 1 and its denominator will be 2.
    After reduction, if the numerator is 0, the method is to make the denominator 1.
- A method called add that receives a Rational object and sums the two objects. The method is to return a new Rational object that contains the reduced sum of the original objects.
  For example, after the following code:

  ```
  Rational r1, r2;
  r1.setNumerator(1);
  r1.setDenominator(6);
  r2.setNumerator(1);
  r2.setDenominator(3);

  Rational r3;
  r3 = r1.add(r2);
  ```

  The values will be:
  r1 will equal 1/6
  r2 will equal 1/3
  r3 will equal 1/2

  Note:
  The argument received is cbv so there is a call to the copy-ctor when the method is called.
- A method called addOne that generates and returns a new Rational object whose value is 1 greater than the value of the calling object.
  For example, given the following code:

  ```
  Rational r1(2,3), r2;
  r2 = r1.addOne();
  ```

  The value of r2 should be 5/3.

Use the main function below in order to check correctness of your class.
The given program checks the methods ctor, copy-ctor, get, set, add, addOne, equals and makeEquals.

The following main function should be used for running the program:

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

```
int main()
{
	int num1, num2, num3;
	char junk;

	cout << "part 1. ctor/set/get" << endl;
	cout << "enter a rational number:" << endl;
	cin >> num1 >> junk >> num2;
	Rational rat1;
	rat1.setNumerator(num1);
	rat1.setDenominator(num2);
	cout << "numerator: " << rat1.getNumerator() << endl;
	cout << "denominator: " << rat1.getDenominator() << endl;
	cout << endl;


	cout << "enter a rational number: " << endl;
	cin >> num1 >> junk >> num2;
	Rational rat2(num1,num2);
	rat2.print();
	cout << endl << endl;

	cout << "part 2. copy-ctor" << endl;
	Rational rat3(rat2);
	rat3.print();
	cout << endl << endl;

	cout << "part 3. equals" << endl;
	if (rat1.equal(rat2))
		cout << "The two numbers are equal" << endl;
	else
		cout << "The two numbers are different" << endl;
	cout << endl;

	cout << "part 4. makeEquals" << endl;
	rat1.makeEquals(rat2);
	if (rat1.equal(rat2))
		cout << "The two numbers are equal" << endl;
	else
		cout << "The two numbers are different" << endl;
	cout << endl;

	cout << "part 5. addOne" << endl;
	Rational ans1;
	ans1 = rat1.addOne();
	rat1.print();
	cout << " + 1 = ";
	ans1.print();
	cout << endl<<endl;

	cout << "part 6. add" << endl;
	Rational ans2;
	ans2 = rat1.add(rat3);
	rat1.print();
	cout << " + ";
```

```
    rat3.print();
    cout << " = ";
    ans2.print();
    cout << endl<<endl;
    return 0;

}
```

Program execution example:

```
part 1. ctor/set/get
enter a rational number:
2/12
print: in constructor
numerator: 2
denominator: 12

enter a rational number:
4/12
print: in constructor
4/12

part 2. copy-ctor
print: in copy-constructor
4/12

part 3. equals
print: in copy-constructor
The two numbers are different

part 4. makeEquals
print: in copy-constructor
The two numbers are equal

part 5. addOne
print: in constructor
print: in constructor
2/12 + 1 = 14/12

part 6. add
print: in constructor
print: in copy-constructor
print: in constructor
2/12 + 4/12 = 1/2
```

Question 2:

Define and implement a class called Employee for calculating the salary of a worker in the fund "Giving from the Heart".

The class must include the following fields:

- The worker id number – a 5 digit int.
- The worker's name – a static array of up to 20 characters: char name[21];
- Hourly wages – a float.
- The number of hours the worker worked – an int.
- The sum of all the donations the worker collected for the fund – a float.

The class must also include the following methods:

- Constructor. The implementation must contain the following output:
  print: in constructor
- For every field:
  - A Setter
  - A Getter
- A method for calculating the salary. The calculation should use the following formula:
  Number of hours * Hourly wages + Percent of donations collected
  The percent factor should be according to the following table:

Donations Collected	Percent Added to Salary
Up to 1000 Shekels (inclusive)	10%
Between 1000 Shekels and 2000 Shekels (inclusive)	15%
Between 2000 Shekels and 4000 Shekels (inclusive)	20%
Between 4000 Shekels and 5000 Shekels (inclusive)	30%
More than 5000 Shekels.	40%

For example: A worker who collected 4500 Shekels for the fund, the percent for his salary will be calculated thus: 1000*0.1+1000*0.15+2000*0.2+500*0.3 = 800.
This means that 800 Shekels will be added to the base salary which is the hourly wages for the hours he worked.

Implement a main program that inputs worker information until the worker id 0 is inputted. For each worker the input will be in the order: worker id, name, wages, hours worked, donations collected. (Assume there is at least one worker).

The program is to print:

- The id and name of the worker who collected the least donations. The sum of collections he collected should also be printed.
- The id and name of the worker with the highest salary. His salary should also be printed.

Note:

- In the case of invalid input, the program should print ERROR and continue to input the data for the current worker.
  Hint: Write the code so that the input of all of the worker's data precedes the validity check.
- Consider the restrictions relevant to each field.
- The main program should not allocate an array of workers.

Program execution examples:

```
print: in constructor
print: in constructor
print: in constructor
enter details, to end enter 0:
12345 moshe   50  40  2000
13579 rivka   120  55  3450
97531 sara  35  100   5632
0
minimum collected: 12345 moshe   2000
highest salary:  13579 rivka 7140
```

```
print: in constructor
print: in constructor
print: in constructor
enter details, to end enter 0:
11111 doron  35  120  6000
22222 tal  50  55  1400
44444 levi  45  -4  100
ERROR
33333 naomi  30  120  800
0
minimum collected: 33333 naomi  800
highest  salary: 11111 doron 5550
```

Good Luck!