CSE225L: Data Structures and Algorithm Lab

Lab 03: Class Template

North South University

Just like function overloading, in C++, it is allowed to overload operators as well. Any operator that is available in C++ can be overloaded except **.** (dot), **::** (scope resolution operator), **?:** and **sizeof**. This is particularly useful when used in a user defined class specially if they represent custom datatypes like fraction, 2d points or complex numbers. Even though we can overload an operator to make it work the way we want; for example: making subtraction operator (-) add two complex numbers, it is recommended to not do so.

In this course, we are only going to study arithmetic operators such as +, -, /, %, * and a few binary logical operators such as ==, !=, <, <=, >, >=.

In the following example of fraction class, we have overloaded the multiplication operator for Fraction class so that we can use it to multiply two fraction objects.

fraction.h

```
#ifndef FRACTION_H
#define FRACTION_H
#include <iostream>

using namespace std;

class Fraction{

private:
    int numerator;
    int denominator;

public:
    Fraction();
    Fraction(int, int);
    int getNumerator();
    void setNumerator(int);
    int getDenominator();
    void setDenominator(int);
    Fraction multiply(Fraction);
    Fraction operator*(Fraction);
    void print();
};

#include "fraction.tpp"
#endif // FRACTION_H
```

fraction.tpp

```
#include "fraction.h"

Fraction::Fraction(){
    numerator = 0;
    denominator = 1;
}
Fraction::Fraction(int numerator, int denominator){
    this->numerator = numerator;
    this->denominator = denominator;
}
void Fraction::print(){

    cout<<numerator<<"/"<<denominator<<endl;

}
int Fraction::getNumerator(){
    return numerator;
}
void Fraction::setNumerator(int numerator){
    this->numerator = numerator;
}
int Fraction::getDenominator(){
    return denominator;
}
void Fraction::setDenominator(int denominator){
    this->denominator = denominator;
}
Fraction Fraction::operator*(Fraction f){
    Fraction result;
    result.numerator = numerator*f.numerator;
    result.denominator = denominator*f.denominator;
    return result;
}
```

Tasks:

1. Modify header and implementation file to overload the following operators in Fraction class.
   i) Logical operators:
   a. **<** (less than),
   b. **>** (greater than),
   c. **==** (logical equal)
   d. **!=** (not equal)
   e. **<=** (less than or equal)
   f. **>=** (greater than or equal)

   ii) Arithmetic operators: **+** (summation), **-** (subtraction) and **/** (division) operators.

   In driver file (main.cpp file), show use of all operators that you have overloaded and print the results.

   Note: You do not need to simplify fractions before or after you obtain your desired results.