CMP 142 – Object Oriented Programming
Fall 2019
LAB-06

Issue Date: October 26, 2020

The objective of this lab is to:

1. Understand and practice unary and binary operators overloading.
2. Practice good coding conventions e.g commenting, meaningful variable and functions names, properly indented and modular code.

Instructions!

1. This is a **graded** lab, you are strictly **NOT** allowed to discuss your solutions with your fellow colleagues, even not allowed asking how is he/she is doing, it may result in negative marking. You can **ONLY** discuss with your TAs or with me.
3. Strictly follow good coding conventions (commenting, meaningful variable and functions names, properly indented and modular code.
4. Save your work frequently. Make a habit of pressing **CTRL+S** after every line of code you write.

Task 01: [20 Marks]

1. Create a class MixdFraction to store a mixed fraction. Recall that a mixed fraction has two parts integral part and fractional part, and is of following form:

3 1/4

```
class MixedFraction
{
private:
      int integer;        // Integral part of the Mixed fraction.
      int numerator;      // Numerator of the fractional part.
      int denominator;    // Denominator of the fractional part. This must be non-zero
public:
      MixedFraction ();  // Initialize data members to default values. Remember
      denominator must not be zero.

      MixedFraction (int a_integer, int a_nmrator, int a_dnmnator);   // Initialize
      data members with parameter values. A mixed number can be negative or positive. If the
      mixed number is to be negative, only integer part can be negative, not numerator or
      denominator.

      MixedFraction (int a_integer); // Constructor with only integer part, set
      fractional part to zero.

      MixedFraction (MixedFraction& a_mFrac); // copy constructor.

      ~ MixedFraction ();     // Destructor should display message “Object is destroyed”.

      void setIntegralPart(int a_intgr);
      bool setDenominator(int a_ dnmnator);
      void setNumerator(int a_ nmrator);
      int getIntegralPart () const;
      int getDenominator () const;
      int getNumerator ()const;

      int evaluate();          // should return the decimal equivalent of the mixed number.
      void display();          // should display the fraction in proper format i.e. 3 2/5.

      MixedFraction operator+(const MixedFraction& ); // overload binary + operator.
      MixedFraction operator-(const MixedFraction& ); // overload binary – operator.
      MixedFraction operator*(const MixedFraction& ); // overload binary * operator.
      MixedFraction operator/(const MixedFraction& ); // overload binary / operator.
```

```
        bool operator==(const MixedFraction& );     // overload relational == operator
        bool operator<(const MixedFraction& );      // overload relational != operator.

        MixedFraction operator-();     // overload unary – operator.
        MixedFraction operator++();    // overload pre-increment operator.
        MixedFraction operator++(int); // overload post-increment operator.

        MixedFraction operator+=(const MixedFraction& ); // overload combined operator +=
        MixedFraction operator/=(const MixedFraction& ); // overload combined operator /=
};
```

Implement the given class declaration. To overload four binary arithmetic operators you should first convert mixed fraction to simple fraction (multiply the integer part by denominator and add that to numerator, write the resultant number as numerator) then perform required operation according to the following rules:

Addition: $$a/b + c/d = (a*d + b*c)/(b*d)$$

Subtraction: $$a/b - c/d = (a*d - b*c)/(b*d)$$

Multiplication: $$a/b * c/d = (a*c)/(b*d)$$

Division: $$\frac{a/b}{c/d} = (a*d)/(b*c)$$

In *main( )* function, create 5 object of MixedFraction and use each function to show your work. Also, display the decimal equivalent of each mixed number.

2. Modify the `MixedFraction (int a_integer)` constructor such that it behaves as default constructor as well and use member initialize list to initialize data members in this constructor.