Object –Oriented Programming

Lab 5 : On Chapter Operator Overloading

ENSIA 2021/2022

Exercise 1

A Rational number can be represented using two fields : **numerator** and **denominator**. Create a class **RationalNumber** (fraction) with the following capabilities :

- **a)** Create a Rational number with specified numerator and denominator or create a default Rational number with numerator 0 and denominator 1, the constructor prevents a 0 denominator in a fraction, reduces or simplifies fractions that are not in reduced form and avoids negative denominators.
- **b)** Overload the addition, subtraction, multiplication and division operators for this class.
- **c)** Oveload the unary operator(-) , the += operator , the preincrement and postincrement (++) operators.
- **d)** Overload the relational and equality operators for this class.
- **e)** Overload the insertion operator (<<) .
- **f)** Convert a rational number into an integer, floating-point value, or string.

Exercise 2

A Polynomial is represented with terms , each term contains a coefficient and an exponent. The term $\mathbf{2x^4}$ has the coefficient 2 and the exponent 4.

The internal representation of a polynomial is an array of coefficients and an array of exponents. According to the template given below develop a complete class **Polynomial**. This class should provide the following oveloaded operators capabilities :

- **a)** Overload the addition operator (**+**) to add two **Polynomials**
- **b)** Overload the subtraction operator (**-**) to subtract two **Polynomials**
- **c)** Overload the assignment operator (**=**) to assign one **Polynomial** to another.
- **d)** Overload the multiplication operator (*) to multiply two **Polynomials** .
- **e)** Overload the addition assignment operator (**+=**), the addition subtraction assignment operator (**-=**) and the multiplication assignment operator (***=**) .

```
Class Polynomial
{
public :
        Polynomial () ;
        Polynomial operator+ (const Polynomial &) ;
        Polynomial operator- (const Polynomial &) ;
        Polynomial operator* (const Polynomial &) ;
        const Polynomial operator= (const Polynomial &) ;
        Polynomial& operator+= (const Polynomial &) ;
        Polynomial& operator-= (const Polynomial &) ;
        Polynomial& operator*= (const Polynomial &) ;
        void  enterTerms(void) ;    //input terms of polynomial
        void  printPolynomial (void) ;

private :
        int exponents[100] ;
        int coefficients [100] ;
        void polynomial Combine(Polynomial &)  //combine common terms
} ;
```