The objective of this lab is to:

Understand and practice unary and binary operators overloading.

Instructions!

1. Please follow PUCIT dress code before coming to the lab. Keep your student identity cards with you.
2. 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.
1. Strictly follow good coding conventions (commenting, meaningful variable and functions names, properly indented and modular code.
2. Save your work frequently. Make a habit of pressing **CTRL+S** after every line of code you write.
3. Beware of **memory leaks** and **dangling pointers**.

Task 01: [40 Marks]

1. Create a class Fraction to store a fraction in the form on denominator and numerator:

```
class Fraction
{
private:
	int numerator;     // Numerator of the fractional part.
	int denominator;   // Denominator of the fractional part. This must be non-zero
	int gcd( int num1, int num2);  // compute and return the GCD of given numbers.

public:
	Fraction (); // Initialize data members to default values. Remember denominator must
not be zero.

	Fraction (int a_nmrator, int a_dnmnator);   // Initialize data members with
	parameter values.

	Fraction (Fraction& a_Frac); // copy constructor.

	~Fraction ();    // Destructor should display message "Object is destroyed".

	bool setDenominator(int a_ dnmnator);
	void setNumerator(int a_ nmrator);
	int getDenominator () const;
	int getNumerator ()const;
	double reduce();      //evaluate and simplify the fraction.

	Fraction operator+(const Fraction& ); // overload binary + operator.
	Fraction operator-(const Fraction& ); // overload binary - operator.
	Fraction operator*(const Fraction& ); // overload binary * operator.
	Fraction operator/(const Fraction& ); // overload binary / operator.
	bool operator==(const Fraction& );    // overload relational == operator
	bool operator<(const Fraction& );     // overload relational < operator.
	bool operator!=(const Fraction& );    // overload relational != operator

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

	friend  ostream& operator<<(ostream&, const Fraction&); // overload << operator.
	friend  istream& operator<<(istream&, Fraction&);        // overload << operator.
```

```
        Fraction operator+=(const MixedFraction& ); // overload combined operator +=
        Fraction operator^(int );                   // overload power operator ^

};
```

Implement the given class declaration. In *main( )* function, create 5 object of Fraction and use each function to show your work.