Object Oriented Programming (IGS2130)

Lab 11

Instructor:
Choonwoo Ryu, Ph.D.

INHA UNIVERSITY

Exercise #1

Hint
08. p13-14

- From the following Fraction class, add overloaded operator << so the following program can run like below:

```
int main() {
    Fraction f1{ 1,2 }, f2{ 3,4 };
    cout << f1 << ", " << f2 << endl;
    return 0;
}
```

```
1/2, 3/4
```

```
class Fraction {
private:
    int m_numerator;
    int m_denominator;
public:
    Fraction(int numerator = 0, int denominator = 1)
        :m_numerator{ numerator }, m_denominator{denominator}
    {}
};
```

```
friend std::ostream& operator<<(std::ostream&, const Fraction&);
```

Exercise #2

Hint
08. p6

- From the code for Ex#1, add overloaded multiplication operator * so the following program can run like below:

```
int main() {
    Fraction f1{ 1,2 }, f2{ 3,4 };
    cout << f1 << ", " << f2 << endl;
    cout << f1 * f2 << endl;
    cout << f1 * 2 << endl;
    cout << 3 * f2 << endl;
    cout << 2 * f1 * f2 * 3 << endl;
    return 0;
}
```

```
1/2, 3/4
3/8
2/2
9/4
18/8
```

```
1/2, 3/4
3/8
1/1
9/4
9/4
```

```
friend Fraction operator*(const Fraction&, const Fraction&);
```

Exercise #3

Hint
08. p6

- From the code for Ex#2, add overloaded addition operator + and subtraction operator - so the following program can run like below:

```
int main() {
    Fraction f1{ 1,2 }, f2{ 3,4 };
    cout << f1 << ", " << f2 << endl;
    cout << f1 << " * " << f2 << " = " << f1 * f2 << endl;
    cout << f1 << " + " << f2 << " = " << f1 + f2 << endl;
    cout << f1 << " - " << f2 << " = " << f1 - f2 << endl;
    cout << 2 << " - " << f2 << " = " << 2 - f2 << endl;
    cout << 2 << " + " << f1 << " = " << 2 + f1 << endl;

    return 0;
}
```

```
1/2, 3/4
1/2 * 3/4 = 3/8
1/2 + 3/4 = 5/4
1/2 - 3/4 = -1/4
2 - 3/4 = 5/4
2 + 1/2 = 5/2
```

```
friend Fraction operator+(const Fraction&, const Fraction&);
friend Fraction operator-(const Fraction&, const Fraction&);
```

Exercise #4

Hint
08. p18-19

- From the code for Ex#3, add overloaded unary negative operator - so the following program can run like below:

```
int main() {
    Fraction f1{ 1,2 }, f2{ 3,4 };
    cout << f1 << ", " << f2 << endl;
    cout << -f1 << ", " << -f2 << endl;
    cout << -(-f1) << endl;

    return 0;
}
```

```
1/2, 3/4
-1/2, -3/4
1/2
```

```
Fraction operator-();
```