Bangladesh University of Business & Technology (BUBT)

Department of Computer Science and Engineering

Assignment - 03: Spring 2023

Course Code: CSE 122 | Course Title: Object Oriented Programming Language Lab

Intake: 50th, Program: B.Sc in CSE (Bi-Semester)

Marks – 10

CO	Question
CO3	**Demonstrate** a C++ code that creates a class called *Fraction*. The class Fraction has two attributes: *numerator and denominator*. • In your *constructor (inyour__init__ method)*, verify(assert?) that the numerator and denominator passed in during initiation are both of type int. If you want to be thorough, also check to make sure that the denominator is not zero. • Write a *.reduce() method* that will reduce a fraction to lowest terms. • Override the Object class's __str__ and __repl__ methods so that your objects will print out nicely. Remember that__str__ is more for humans; __repl__ is more for programmers. Ideally ,the__repl__ method will produce a string that you can run through the eval() function to clone the original fraction object. • Override the + operator. In your code, this means that you will implement the special method __add__. The signature of the __add__ function will be def __add__(self, other): , and you'll return a new Fraction with the result of the addition. Run your new Fraction through the *reduce() function* before returning.

Here is the answer of this question ----

```
#include <iostream>
#include <conio.h>
#include <cmath>

using namespace std;
```

```
  class Fraction { private:
    int numerator;
   int denominator;
  public:
    Fraction(int num, int denom) {
      assert(denom != 0);
        assert(typeid(num) == typeid(int) && typeid(denom) == typeid(int));
numerator = num;
  denominator = denom;
    }
    void reduce() {
        int gcd = (numerator, denominator);

        numerator /= gcd;
   denominator /= gcd;
         if (denominator < 0) {
    numerator = -numerator;
            denominator = abs(denominator);
        }
    }

    Fraction operator+(Fraction const &f2) {
        int new_num = numerator * f2.denominator + f2.numerator *
denominator;
        int new_denom = denominator * f2.denominator;
        Fraction result(new_num, new_denom);
        result.reduce();
    return result;
    }
    friend ostream& operator<<(ostream& os, const Fraction& f) {
  os << f.numerator << "/" << f.denominator;
    return os;
    }
    string repr() const {
        return "Fraction(" + to_string(numerator) + ", " +
to_string(denominator) + ")";
```

```
    }
};
int main() {
    Fraction f1(3, 4);
    Fraction f2(1, 2);
    Fraction f3 = f1 + f2;
      cout << "f1 = " << f1 << endl;
   cout << "f2 = " << f2 << endl;
  cout << "f3 = " << f3 << endl;
    cout << "f3.repr() = " << f3.repr() << endl;

    return 0;
}
```