Bangladesh University of Business and Technology

BUBT

Assignment

On

Course Title : Object Oriented Programming

Course Code : CSE 122

Submitted to :

Khan Md. Hasib

Lecturer

Department of CSE

BUBT

Submitted by :

Name : Faria Akther Meghla

ID : 21225103120

Intake : Intake-49

Sec : 03

Assignment 3

Question : 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

Answer:

```
#include <iostream>

#include <cassert>


using namespace std;


class Fraction {

private:

   int numerator;

   int denominator;

public:

   Fraction(int n, int d) {
```

```
    assert(d != 0);
    assert(typeid(n) == typeid(int));
    assert(typeid(d) == typeid(int));
    numerator = n;
    denominator = d;
}

void reduce() {
    int gcd = findGCD(numerator, denominator);
    numerator /= gcd;
    denominator /= gcd;
}

int findGCD(int a, int b) {
    if (b == 0) {
        return a;
    }
    return findGCD(b, a % b);
}

Fraction operator+(const Fraction &other) const {
    int num = numerator * other.denominator + other.numerator * denominator;
    int den = denominator * other.denominator;
    Fraction result(num, den);
```

```
        result.reduce();
        return result;
    }

    friend ostream &operator<<(ostream &output, const Fraction &frac) {
        output << frac.numerator << "/" << frac.denominator;
        return output;
    }

    string repr() const {
        return "Fraction(" + to_string(numerator) + ", " + to_string(denominator) +
")";
    }
};

int main() {
    Fraction a(3, 4);
    Fraction b(1, 2);
    Fraction c = a + b;
    cout << a << " + " << b << " = " << c << endl;
    cout << "Representation of c: " << c.repr() << endl;
    return 0;
}
```