Page |

1



**ASSIGNMENT**

**Assignment no -01**

Course NO

Course Name

Submission Date

:CSE 122

:

Object Oriented Programming language Lab

:10-04-2023

**Submitted To**

**Name: Khan Md. Hasib**

Assistant Professor

Department of Computer Science & Engineering

**Submitted By**

**Name: Md.Sakib Abdullah**

**ID:22234103230 INATKE:**

50

**SECTION:06**

C 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.

SOLUTIOONS:

#include <iostream>

#include <cassert>

#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) + ")";

}

Page | 4

};

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;

}