Jaypee University of Engineering an Technology, Guna
Department of Computer Science and Engineering
Object Oriented Programming Lab(18B17CI271) -6

Submitted by :- Mohammed Raza Khan
Enrolment no. :- 201B156
Batch :- BX2(B5)

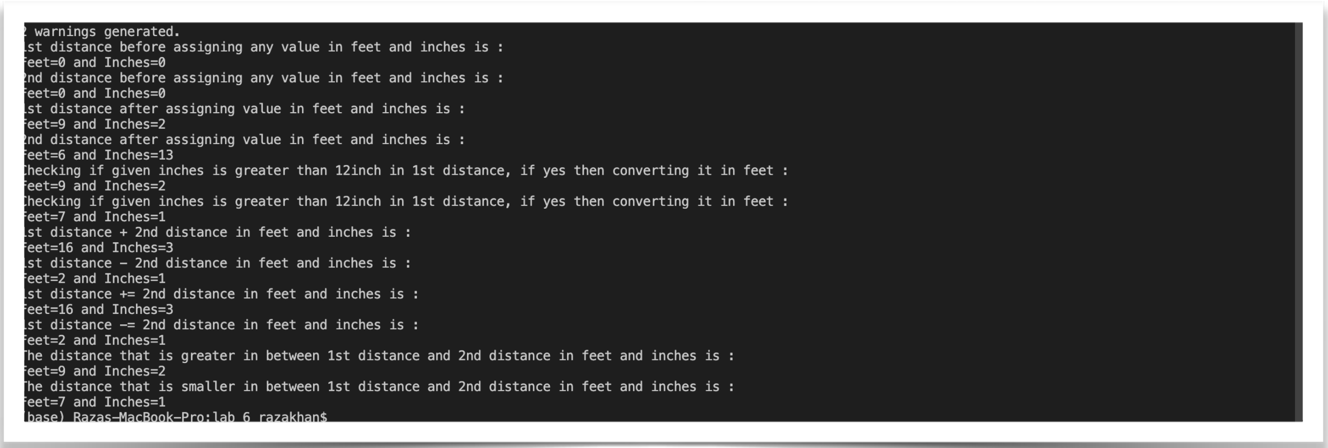
1. Design a class Distance that includes following data members: feet, inches. It has the following member function:-
• Constructor, that initializes the distance to 0,0 by default.
• Give a check so that the inches part is always less than 12.0. • Display function
• Overloaded – operator to subtract 2 distances
• Overloaded + operator to add 2 distances
• Overload += and -= operator
• Overload > and < operators to compare two distances

Code:

```
 #include <iostream>
using namespace std;
class Distance
{
    int feet, inches;

public:
    Distance()
    {
        feet = 0;
        inches = 0;
    }
    void getinput(int feet, int inches)
    {
        this->feet = feet;
        this->inches = inches;
    }
    int check()
    {
        if (inches > 12)
        {
            feet = feet + (inches / 12);
            inches = inches % 12;
        }
        else
        {
            inches = inches;
        }
    }
    void display()
    {
        cout << "Feet=" << feet << " and Inches=" << inches << endl;
    }
    Distance operator-(Distance obj)
    {
        Distance temp_obj;
    temp_obj.feet = feet - obj.feet;
    temp_obj.inches = inches - obj.inches;
    return (temp_obj);
}
Distance operator+(Distance obj)
{
    Distance temp_obj;
    temp_obj.feet = feet + obj.feet;
    temp_obj.inches = inches + obj.inches;
    return (temp_obj);
}
friend Distance operator+=(Distance obj1, Distance obj2)
{
    obj1.feet += obj2.feet;
    obj1.inches += obj2.inches;
    return obj1;
}
friend Distance operator-=(Distance obj1, Distance obj2)
{
    obj1.feet -= obj2.feet;
    obj1.inches -= obj2.inches;
    return obj1;
}
Distance operator>(Distance obj)
{
    Distance temp_obj;
    if (feet > obj.feet)
    {
        temp_obj.feet = feet;
    }
    else
    {
        temp_obj.feet = obj.feet;
    }
    if (inches > obj.inches)
    {
        temp_obj.inches = inches;
    }
    else
    {
        temp_obj.inches = obj.inches;
    }
    return (temp_obj);
}
Distance operator<(Distance obj)
{
    Distance temp_obj;
    if (feet < obj.feet)
    {
        temp_obj.feet = feet;
    }
    else
    {
        temp_obj.feet = obj.feet;
    }
    if (inches < obj.inches)
    {
        temp_obj.inches = inches;
    }
    else
```

```
        {
            temp_obj.inches = obj.inches;
        }
        return (temp_obj);
    }
};
int main()
{
    Distance obj1, obj2, obj3;
    cout << "1st distance before assigning any value in feet and inches is : " << endl;
    obj1.display();
    cout << "2nd distance before assigning any value in feet and inches is : " << endl;
    obj2.display();
    obj1.getinput(9, 2);
    cout << "1st distance after assigning value in feet and inches is : " << endl;
    obj1.display();
    obj2.getinput(6, 13);
    cout << "2nd distance after assigning value in feet and inches is : " << endl;
    obj2.display();
    obj2.check();
    obj1.check();
    cout << "Checking if given inches is greater than 12inch in 1st distance, if yes then converting it
in feet : " << endl;
    obj1.display();
    cout << "Checking if given inches is greater than 12inch in 1st distance, if yes then converting it
in feet : " << endl;
    obj2.display();
    obj3 = obj1 + obj2;
    cout << "1st distance + 2nd distance in feet and inches is : " << endl;
    obj3.display();
    obj3 = obj1 - obj2;
    cout << "1st distance - 2nd distance in feet and inches is : " << endl;
    obj3.display();
    obj3 = (obj1 += obj2);
    cout << "1st distance += 2nd distance in feet and inches is : " << endl;
    obj3.display();
    obj3 = (obj1 -= obj2);
    cout << "1st distance -= 2nd distance in feet and inches is : " << endl;
    obj3.display();
    obj3 = obj1 > obj2;
    cout << "The distance that is greater in between 1st distance and 2nd distance in feet and
inches is : " << endl;
    obj3.display();
    obj3 = obj1 < obj2;
    cout << "The distance that is smaller in between 1st distance and 2nd distance in feet and
inches is : " << endl;
    obj3.display();
    return 0;
}
```



```
 warnings generated.
st distance before assigning any value in feet and inches is :
eet=0 and Inches=0
nd distance before assigning any value in feet and inches is :
eet=0 and Inches=0
st distance after assigning value in feet and inches is :
eet=9 and Inches=2
nd distance after assigning value in feet and inches is :
eet=6 and Inches=13
hecking if given inches is greater than 12inch in 1st distance, if yes then converting it in feet :
eet=9 and Inches=2
hecking if given inches is greater than 12inch in 1st distance, if yes then converting it in feet :
eet=7 and Inches=1
st distance + 2nd distance in feet and inches is :
eet=16 and Inches=3
st distance - 2nd distance in feet and inches is :
eet=2 and Inches=1
st distance += 2nd distance in feet and inches is :
eet=16 and Inches=3
st distance -= 2nd distance in feet and inches is :
eet=2 and Inches=1
he distance that is greater in between 1st distance and 2nd distance in feet and inches is :
eet=9 and Inches=2
he distance that is smaller in between 1st distance and 2nd distance in feet and inches is :
eet=7 and Inches=1
base) Razas-MacBook-Pro:lab 6 razakhan$
```

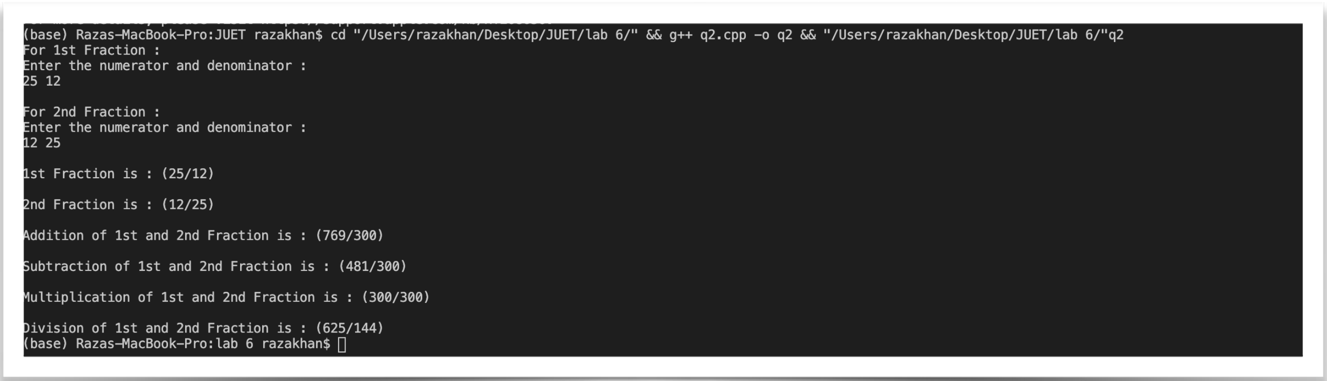
Q2. Create a class rational for performing arithmetic with fractions. Use an integer variable to represent the private data of the class-the numerator and denominator. Provide a member function to get input from the user. This function should also check that denominator entered is not 0, if it is zero print invalid input. Provide a function to display the values. Overload +, -, *, / operators to add, subtract, multiply and divide the objects of this class.

Code:
```
#include <iostream>
using namespace std;
class rational
{
    int numerator, denominator;
public:
    void getinput()
    {
        cout << "Enter the numerator and denominator :" << endl;
        cin >> numerator >> denominator;
        if (denominator == 0)
        {

            cout << "Invalid Input : "
                 << "(" << numerator << "/" << denominator << ")" << endl;
        }
    }
    void display()
    {
        cout << "(" << numerator << "/" << denominator << ")" << endl;
    }
    rational operator+(rational obj1)
    {
        rational temp_obj;
        temp_obj.numerator = (numerator * (obj1.denominator) + denominator * (obj1.numerator));
        temp_obj.denominator = (denominator * (obj1.denominator));
        return temp_obj;
    }
    rational operator-(rational obj1)
    {
        rational temp_obj;
        temp_obj.numerator = (numerator * (obj1.denominator) - denominator * (obj1.numerator));
        temp_obj.denominator = (denominator * (obj1.denominator));
        return temp_obj;
    }
    rational operator*(rational obj1)
    {
        rational temp_obj;
        temp_obj.numerator = (numerator * (obj1.numerator));
        temp_obj.denominator = (denominator * (obj1.denominator));
        return temp_obj;
    }
    rational operator/(rational obj1)
    {
        rational temp_obj;
        temp_obj.numerator = (numerator * (obj1.denominator));
        temp_obj.denominator = (denominator * (obj1.numerator));
        return temp_obj;
    }
```

```cpp
};
int main()
{
    rational obj1, obj2, obj3;
    cout << "For 1st Fraction :" << endl;
    obj1.getinput();
    cout << endl
         << "For 2nd Fraction :" << endl;
    obj2.getinput();
    cout << endl
         << "1st Fraction is : ";
    obj1.display();
    cout << endl
         << "2nd Fraction is : ";
    obj2.display();
    cout << endl
         << "Addition of 1st and 2nd Fraction is : ";
    obj3 = obj1 + obj2;
    obj3.display();
    cout << endl
         << "Subtraction of 1st and 2nd Fraction is : ";
    obj3 = obj1 - obj2;
    obj3.display();
    cout << endl
         << "Multiplication of 1st and 2nd Fraction is : ";
    obj3 = obj1 * obj2;
    obj3.display();
    cout << endl
         << "Division of 1st and 2nd Fraction is : ";
    obj3 = obj1 / obj2;
    obj3.display();
    return 0;
}
```

Output:



```
(base) Razas-MacBook-Pro:JUET razakhan$ cd "/Users/razakhan/Desktop/JUET/lab 6/" && g++ q2.cpp -o q2 && "/Users/razakhan/Desktop/JUET/lab 6/"q2
For 1st Fraction :
Enter the numerator and denominator :
25 12

For 2nd Fraction :
Enter the numerator and denominator :
12 25

1st Fraction is : (25/12)

2nd Fraction is : (12/25)

Addition of 1st and 2nd Fraction is : (769/300)

Subtraction of 1st and 2nd Fraction is : (481/300)

Multiplication of 1st and 2nd Fraction is : (300/300)

Division of 1st and 2nd Fraction is : (625/144)
(base) Razas-MacBook-Pro:lab 6 razakhan$ 
```

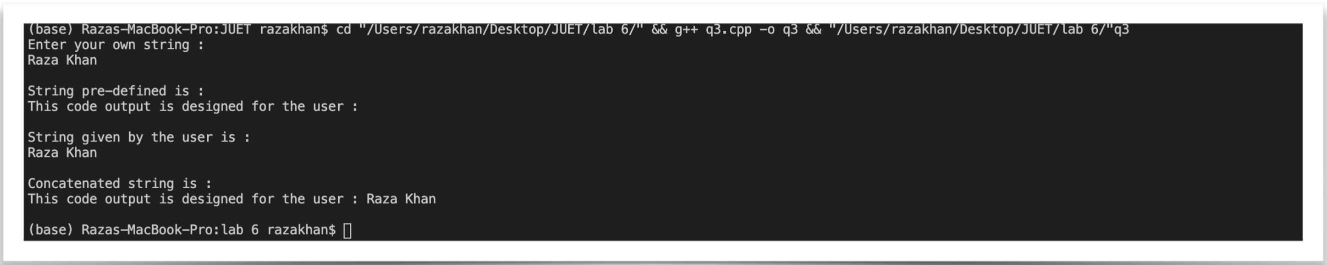
Q3 Include a function that adds two strings to make a third string. Write a program to do the following tasks:
i. Create uninitialized string objects
ii. Creates the objects with string constants. iii. Concatenates two strings properly.
iv. Displays a desired string object

Code:

```cpp
#include <iostream>
using namespace std;
class concatenate
{
    string s;

public:
    void setstring(string s)
    {
        this->s = s;
    }
    void user_setstring()
    {
        cout << "Enter your own string : " << endl;
        getline(cin, s);
    }
    void display()
    {

        cout << s << endl
             << endl;
    }
    friend concatenate operator+(concatenate obj1, concatenate obj2)
    {
        concatenate temp_obj;
        temp_obj.s = (obj1.s).append(obj2.s);
        return temp_obj;
    }
};
int main()
{
    concatenate obj1, obj2, obj3;
    obj1.setstring("This code output is designed for the user : ");
    obj2.user_setstring();
    cout << endl
         << "String pre-defined is : " << endl;
    obj1.display();
    cout << "String given by the user is : " << endl;
    obj2.display();
    obj3 = obj1 + obj2;
    cout << "Concatenated string is : " << endl;
    obj3.display();
    return 0;
}
```



```
(base) Razas-MacBook-Pro:JUET razakhan$ cd "/Users/razakhan/Desktop/JUET/lab 6/" && g++ q3.cpp -o q3 && "/Users/razakhan/Desktop/JUET/lab 6/"q3
Enter your own string :
Raza Khan

String pre-defined is :
This code output is designed for the user :

String given by the user is :
Raza Khan

Concatenated string is :
This code output is designed for the user : Raza Khan

(base) Razas-MacBook-Pro:lab 6 razakhan$ 
```