Assignment 5

**Question 1:** Write a program to create a class Complex. This class will

contain 3 constructors :

(i.) Default (ii.) With one parameter which will take some value for real and imaginary

part of a complex number. (iii.) Two parameters having different values of real and

imaginary part.

The program will have 2 functions :

(i.) Add two complex number by taking two reference variable of class complex which

will return another reference.

(ii.) Display the Result.

**Code:**

#include <iostream>

class Complex

{

public:

double real, img;

Complex()

{

real = 0; img = 0;

}

Complex(double x)

{

real = x; img = 0;

}

Complex(double x, double y)

{

real = x; img = y;

}

Complex operator+(const Complex &b)

{

return Complex(real+b.real, img + b.img);

}

};

std::ostream& operator<<(std::ostream &out, Complex a)

{

out << a.real << " + " << a.img << "i";

return out;

}

int main()

{

Complex a(1, 2), b(3, 4);

a = a + b;

std::cout << "Ans: " << a << '\n';

}

**Output:**



**Question 2:** Create a class/struct student having record of name and roll no and then

sort them in increasing order in terms of their alphabetical precedence in case the two names are same compare them in order of their roll no.

**Code:**

#include <iostream>

#include <vector>

#include <algorithm>

struct Student

{

std::string name; int roll;

Student(std::string x, int y)

{

name = x; roll = y;

}

};

int main()

{

int n; std::cin>>n;

std::vector<Student> arr(n, Student("", 0));

for(auto& i : arr)

std::cin >> i.name >> i.roll;

std::sort(arr.begin(), arr.end(), [](const Student &x, const Student &y){

if(x.name != y.name)

return x.name < y.name;

return x.roll < y.roll;

});

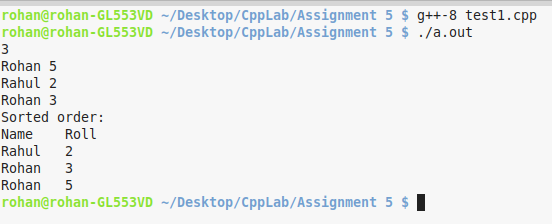
std::cout << "Sorted order:\nName\tRoll\n";

for(const auto& i : arr)

std::cout << i.name <<'\t' << i.roll << '\n';

}

**Output:**



**Question 3:** Write a program to declare two 2d vectors v1 and v2. Set the

content of v1 = v2 by overloading the = operator. Finally check whether the content of two 2d vectors are equal or not by overloading the == operator.

**Code:**

#include <iostream>

#include <vector>

struct Vector

{

std::vector<std::vector<int>> arr;

void operator=(const Vector &x)

{

for(int i = 0; i < x.arr.size(); i++)

for(int j = 0; j < x.arr[0].size(); j++)

arr[i][j] = x.arr[i][j];

}

bool operator==(const Vector &x)

{

for(int i = 0; i < x.arr.size(); i++)

for(int j = 0; j < x.arr[0].size(); j++)

if(arr[i][j] != x.arr[i][j])

return false;

return true;

}

};

int main()

{

Vector c1, d1;

c1.arr = {{1, 2, 3}, {4, 5, 6}};

d1.arr = {{4, 5, 6}, {1, 2, 3}};

std::cout << (c1 == d1) << '\n';

d1 = c1;

std::cout << (c1 == d1) << '\n';

}

**Output:**

