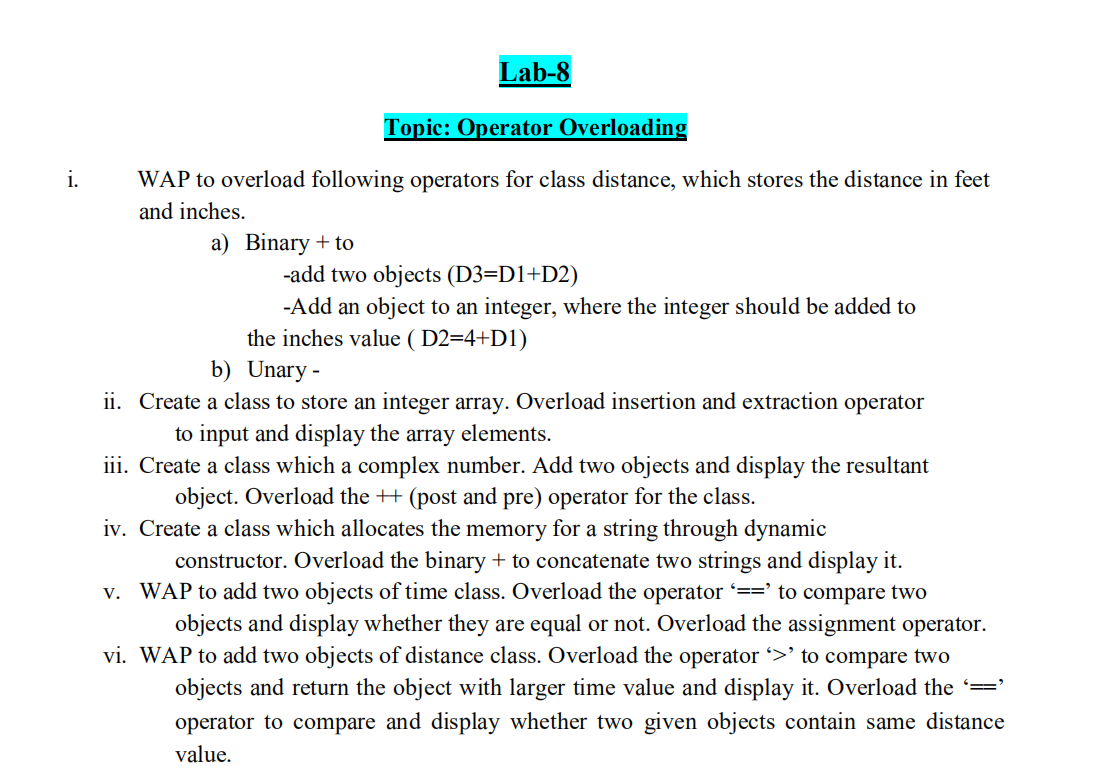
## **CHAUDHARY HAMDAN**

**1905387**

**OOP LAB-8**

**Date : 16-10-2020**



**1.**

#include <iostream>

using namespace std;

class Distance {

private:

int feet;

int inches;

public:

Distance() {

feet = 0;

inches = 0;

}

Distance(int f, int i) {

feet = f;

inches = i;

}

friend ostream &operator<<( ostream &output, const Distance &D ) {

output << "F : " << D.feet << " I : " << D.inches;

return output;

}

friend istream &operator>>( istream &input, Distance &D ) {

input >> D.feet >> D.inches;

return input;

}

//operator overlaoading

Distance operator +(Distance d){

Distance temp;

temp.feet = feet+d.feet;

temp.inches = inches+d.inches;

return temp;

}

};

int main() {

Distance D1(11, 10), D2(5, 11), D3, D4;

cout << "Enter the value of object : " << endl;

cin >> D3;

cout << "First Distance : " << D1 << endl;

cout << "Second Distance :" << D2 << endl;

cout << "Third Distance :" << D3 << endl;

cout<<"Add : d1 and d2 -"<<endl;

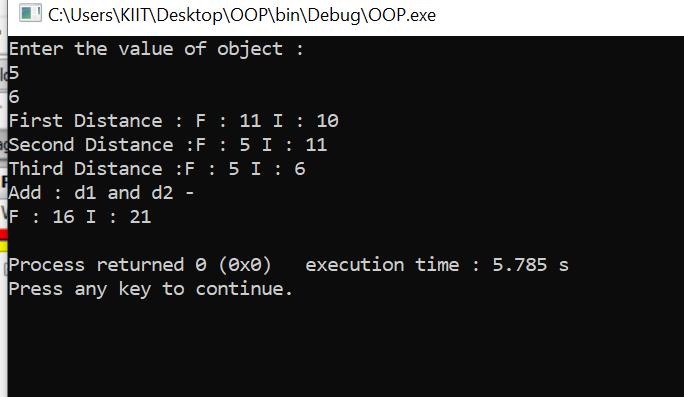
D4 = D1 + D2;

cout<<D4<<endl;

return 0;

}

**OUTPUT :**



**2.**

#include<iostream>

using namespace std;

class A

{ public:

int a[5];

friend istream& operator>>(istream &din,A &ob);

friend ostream& operator<<(ostream &dout,A &ob);

};

istream& operator>>(istream &din,A &ob)

{

for(int i=0;i<5;i++)

{

din>>ob.a[i];

}

return din;

}

ostream& operator<<(ostream &dout,A &ob)

{

for(int i=0;i<5;i++)

{

dout<<ob.a[i]<<" ";

}

return dout;

}

int main()

{

A obj;

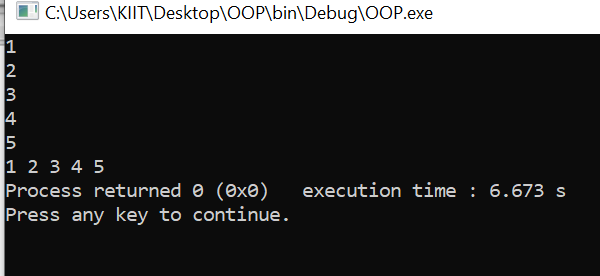
cin>>obj;

cout<<obj;

return 0;

}

**OUTPUT :**



**3.**

#include<bits/stdc++.h>

using namespace std;

class Complex{

private:

int real, img;

public:

Complex(){}

Complex(int r, int i){

real = r; img = i;

}

void print(){

cout<<real<<" + "<<img<<"i"<<endl;

}

void operator ++(){

++real;

}

//operator overlaoading

Complex operator +(Complex c){

Complex temp;

temp.real = real+c.real;

temp.img = img+c.img;

return temp;

}

};

int main(){

Complex c1(5, 4);

Complex c2(2, 5);

Complex c3;

c3 = c1 + c2;//c3 = c1.add(c2);

//cout<<c3<<endl;

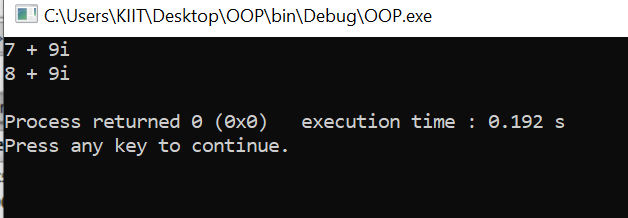
c3.print();

++c3;

c3.print();

}

**OUTPUT :**



4.

#include<iostream>

#include<string.h>

using namespace std;

class String

{

private:

int length;

char \*str;

public:

String(){

length=0;

str= new char[length+1];

}

String(char \*s){

length=strlen(s);

str=new char[length+1];

strcpy(str,s);

}

friend String operator +(String &s1, String &s2);

friend void show(String &s);

};

void show(String &s){

cout<<s.str<<endl;

}

String operator +(String &s1, String &s2){

String temp;

temp.length=s1.length+s2.length;

delete temp.str;

temp.str=new char[temp.length+1];

strcpy(temp.str,s1.str);

strcat(temp.str,s2.str);

return temp;

}

int main()

{

String s1("1905387\_Chaudhary "), s2("Hamdan");

String s3;

s3=s1+s2;

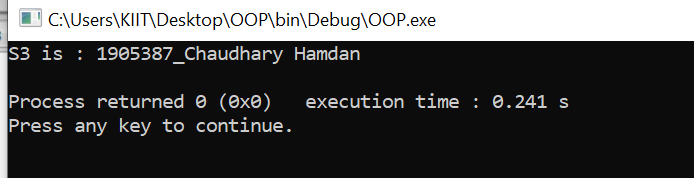
cout<<"S3 is : ";

show(s3);

return 0;

}

**OUTPUT :**

****

**5.**

#include <iostream>

using namespace std;

class time{

int hr,min;

public:

time()

{}

time(int r, int i){

hr=r;

min=i;

}

time operator+(time c3)

{

time c;

c.hr=hr+c3.hr;

c.min=min+c3.min;

return c;

}

int operator==(time c1){

if(hr == c1.hr && min == c1.min)

return 1;

else

return 0;

}

void show()

{

cout<<hr<<","<<min;

}

void operator = (time b)

{

hr=b.hr;

min=b.min;

}

};

int main() {

time c1(5,5),c2(5,5),c3(5,6),c4;

cout<<"\nequal to operator over loading\n";

if((c1==c2)==1)cout<<"True\n";

else cout<<"False\n";

if((c1==c3)==1)cout<<"True\n";

else cout<<"False\n";

cout<<"\n\nassignment operator overloading\n";

c1=c3;

c1.show();

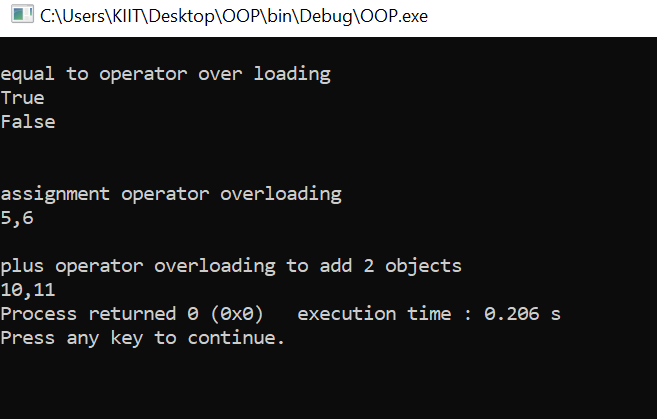
cout<<"\n\nplus operator overloading to add 2 objects\n";

c4=c2+c3;

c4.show();

}

**OUTPUT :**

****

**6.**

#include<iostream>

using namespace std;

class dist

{

float feet, inches;

public:

dist( ){

feet=inches=0.0;

}

dist (float f, float i)

{

feet=f;

inches=i;

}

bool operator > (dist d2);

bool operator==(dist d2);

dist operator + (dist d2);

void display()

{

cout<<feet<<"feet "<<inches<<"inches"<<"\n";

}

};

dist dist:: operator+(dist d2)

{

dist d;

d.feet= feet + d2.feet;

d.inches=inches+d2.inches;

return d;

}

bool dist:: operator >(dist d2)

{ float t1, t2;

t1= feet + inches/12.0;

t2= d2.feet + d2.inches/12.0;

return (t1>t2)? true : false;

}

bool dist:: operator==(dist d2){

if(feet == d2.feet && inches == d2.inches)

return true;

else

return false;

}

int main()

{

dist d1(5,7), d2(7,11), d3(5,7),d4;

cout<<"\ngreater than operator over loading\n";

if (d1 > d2)

cout<<"dist1 is more \n";

else

cout<<"dist1 is less than dist2 \n";

cout<<"\nequal to operator over loading\n";

if (d1 == d3)

cout<<"equal \n";

else

cout<<"unequal \n";

cout<<"\n\nplus operator overloading to add 2 objects\n";

d4=d2+d3;

d4.display();

}

**OUTPUT :**

