**Cisco Tasks**

**LAB #** **03**

**Fall 2019**

**CSE208L Object Oriented Programming Lab**

Submitted by: **Shah Raza**

Registration No. : **18PWCSE1658**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr. Sumayyea Salahuddin**

November 20, 2019

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

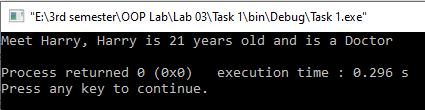
# Activity # 01

**Title:**

Person Class.

**In C++**

**Source code: Output:**



#include <iostream>

#include <string>

using namespace std;

class Person

{

public:

string Name;

int age;

string occupation;

string name(){return Name;}

};

void print(Person\* person)

{

cout<<", "<<person->Name<<" is "<<person->age<<" years old and is a "<<person->occupation<<endl;

}

int main()

{

Person person;

person.Name="Harry";

person.age=21;

person.occupation="Doctor";

cout<<"Meet "<<person.name();

print(&person);

return 0;

}

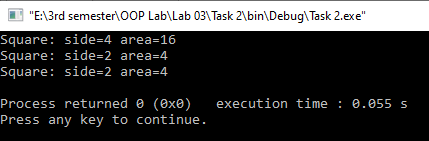
# Activity # 02

**Title:**

Square Class.

**In C++**

**Source code: Output:**



#include <iostream>

#include <string>

using namespace std;

class Square

{

public:

Square (double side);

void set\_side(double side)

{

if (side>=0)

{

this->side=side;

this->area=side\*side;

}

}

void print();

private:

double side;

double area;

};

Square::Square(double side)

{

this->side=side;

this->area=side\*side;

}

void Square::print()

{

cout<<"Square: side="<<side<<" area="<<area<<endl;

}

int main()

{

Square s(4);

s.print();

s.set\_side(2.0);

s.print();

s.set\_side(-33.0);

s.print();

return 0;

}

# Activity # 03

**Title:**

AdHocSquare Class.

**In C++**

**Source code:**

#include <iostream>

using namespace std;

class AdHocSquare

{

public:

AdHocSquare(double side);

void set\_side(double side);

double get\_area();

private:

double side;

};

AdHocSquare::AdHocSquare(double side)

{

this->side=side;

}

void AdHocSquare::set\_side(double side)

{

if (side>=0)

this->side=side;

}

double AdHocSquare::get\_area()

{

return side\*side;

}

class LazySquare

{

public:

LazySquare(double side);

void set\_side(double side);

double get\_area();

private:

double side;

double area;

bool side\_changed;

};

LazySquare::LazySquare(double side)

{

this->side=side;

this->area=side\*side;

}

void LazySquare::set\_side(double side)

{

if (side>=0)

{

if (side!=this->side)

side\_changed=true;

else

side\_changed=false;

this->side=side;

}

}

double LazySquare::get\_area()

{

if (side\_changed)

{

area=side\*side;

}

return area;

}

int main()

{

AdHocSquare A(8);

LazySquare B(19);

cout<<"A: Area= "<<A.get\_area()<<endl;

A.set\_side(3);

cout<<"A: Area= "<<A.get\_area()<<endl;

cout<<"B: Area= "<<B.get\_area()<<endl;

B.set\_side(19);

cout<<"B: Area= "<<B.get\_area()<<endl;

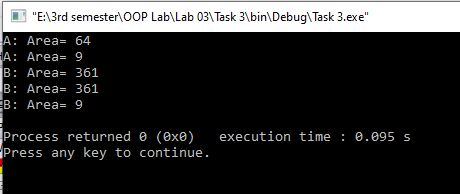
B.set\_side(3);

cout<<"B: Area= "<<B.get\_area()<<endl;

return 0;

}

**Output:**



# Activity # 04

**Title:**

ShopItemOrder Class.

**In C++**

**Source code:**

#include <iostream>

#include<string>

using namespace std;

class ShopItemOrder

{

private:

string name;

double unit\_price;

int items\_ordered;

public:

void get()

{

cout<<"Enter name of the item: ";

getline(cin,name);

cout<<"Enter Unit Price: ";

cin>>unit\_price;

cout<<"Enter Number of Orders: ";

cin>>items\_ordered;

}

double total\_price()

{

return unit\_price\*items\_ordered;

}

void print()

{

cout<<"Item name: "<<name<<endl;

cout<<"Unit price: "<<unit\_price<<endl;

cout<<"Items Ordered: "<<items\_ordered<<endl;

cout<<"Total Price: "<<total\_price();

}

};

int main()

{

ShopItemOrder item;

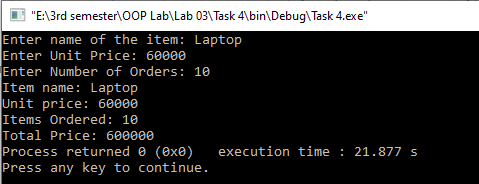
item.get();

item.print();

return 0;

}

**Output:**



# Activity # 05

**Title:**

FlightBooking Class.

**In C++**

**Source code:**

#include <iostream>

using namespace std;

class FlightBooking

{

public:

FlightBooking(int id, int capacity, int reserved);

void printStatus();

private:

int id;

int capacity;

int reserved;

};

void FlightBooking::printStatus()

{

cout<<"Flight "<<id<<" : "<<reserved<<"/"<<capacity<<" ( "<<(float(reserved)/float(capacity))\*100<<"% ) seats taken"<<endl;

}

FlightBooking::FlightBooking(int id, int capacity, int reserved)

{

this->id=id;

this->capacity=capacity;

this->reserved=reserved;

}

int main()

{

int reserved=0,capacity=0;

cout<<"Provide Flight Capacity: ";

cin>>capacity;

cout<<"Provide Number of Reserved seats: ";

cin>>reserved;

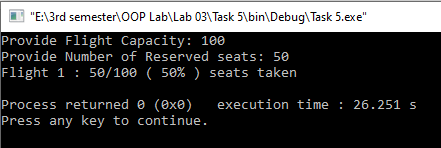
FlightBooking booking(1,capacity,reserved);

booking.printStatus();

return 0;

}

**Output:**



# Activity # 06

**Title:**

FlightBooking:Part 2 Class.

**In C++**

**Source code:**

#include <iostream>

#include<string>

#include<stdio.h>

#include<stdlib.h>

using namespace std;

class FlightBooking

{

public:

FlightBooking(int id, int capacity, int reserved);

void printStatus();

bool reserveSeats(int number\_ob\_seats);

bool cancelReservations(int number\_ob\_seats);

private:

int id;

int capacity;

int reserved;

};

void FlightBooking::printStatus()

{

cout<<"Flight "<<id<<" : "<<reserved<<"/"<<capacity<<" ( "<<(float(reserved)/float(capacity))\*100<<"% ) seats taken"<<endl;

}

bool FlightBooking::reserveSeats(int number\_ob\_seats)

{

if ((float(reserved+number\_ob\_seats)/float(capacity))\*100<=105)

{

reserved+=number\_ob\_seats;

return true;

}

return false;

}

bool FlightBooking::cancelReservations(int number\_ob\_seats)

{

if(number\_ob\_seats<=reserved)

{

reserved-=number\_ob\_seats;

return true;

}

return false;

}

FlightBooking::FlightBooking(int id, int capacity, int reserved)

{

this->id=id;

this->capacity=capacity;

if (reserved>=0)

{

if((float(reserved)/float(capacity))\*100<=105)

this->reserved=reserved;

else

cout<<"Can't reserve more than 105%.\n";

}

else

this->reserved=0;

}

int main()

{

int reserved=0,capacity=0;

cout<<"Provide Flight Capacity: ";

cin>>capacity;

cout<<"Provide Number of Reserved seats: ";

cin>>reserved;

FlightBooking booking(1,capacity,reserved);

string command="";

while(command!="quit")

{

booking.printStatus();

cout<<"What would you like to do?: ";

cin>>command;

if(command[0]=='a')

{

string s= command.substr(3);

int n=stoi(s);

if(!booking.reserveSeats(n))

cout<<"Can't perform this action.\n";

}

else

{

string s=command.substr(6);

int n=stoi(s);

if (!booking.cancelReservations(n))

cout<<"Can't perform this action.\n";

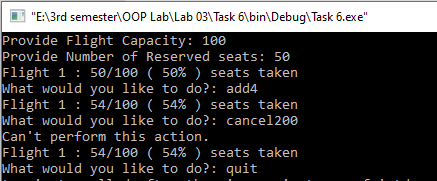
}

}

return 0;

}

**Output:**



# Activity # 07

**Title:**

FlightBooking:Part 3 Class.

**In C++**

**Source code:**

#include <iostream>

#include<string>

#include<stdio.h>

#include<stdlib.h>

using namespace std;

class FlightBooking

{

public:

FlightBooking(int id, int capacity, int reserved);

FlightBooking();

void printStatus();

bool reserveSeats(int number\_ob\_seats);

bool cancelReservations(int number\_ob\_seats);

bool set\_id(int num);

void set\_capacity(int num);

int get\_id(){return id;}

private:

int id;

int capacity;

int reserved;

};

FlightBooking::FlightBooking()

{

id=0;capacity=0;reserved=0;

}

bool FlightBooking::set\_id(int num)

{

id=num;

return true;

}

void FlightBooking::set\_capacity(int num)

{

capacity=num;

}

void FlightBooking::printStatus()

{

cout<<"Flight "<<id<<" : "<<reserved<<"/"<<capacity<<" ( "<<(float(reserved)/float(capacity))\*100<<"% ) seats taken"<<endl;

}

bool FlightBooking::reserveSeats(int number\_ob\_seats)

{

if ((float(reserved+number\_ob\_seats)/float(capacity))\*100<=105)

{

reserved+=number\_ob\_seats;

return true;

}

return false;

}

bool FlightBooking::cancelReservations(int number\_ob\_seats)

{

if(number\_ob\_seats<=reserved)

{

reserved-=number\_ob\_seats;

return true;

}

return false;

}

FlightBooking::FlightBooking(int id, int capacity, int reserved)

{

this->id=id;

this->capacity=capacity;

if (reserved>=0)

{

if((float(reserved)/float(capacity))\*100<=105)

this->reserved=reserved;

else

cout<<"Can't reserve more than 105%.\n";

}

else

this->reserved=0;

}

int main()

{

FlightBooking booking[10];

int id,cap,re;

booking[0]= FlightBooking(1,400,0);

string command="";

while(command!="quit")

{

for(int i=0;i<10;i++)

{

if(booking[i].get\_id()!=0)

booking[i].printStatus();

}

cout<<"\nWhat would you like to do?: ";

cin>>command;

if(command=="create")

{

cin>>id>>cap;

for (int i=0;i<10;i++)

{

if(booking[i].get\_id()==0)

{

booking[i].set\_id(id);

booking[i].set\_capacity(cap);

break;

}

}

}

else if(command=="delete")

{

cin>>id;

for(int i=0;i<10;i++)

{

if(booking[i].get\_id()==id)

{

booking[i].set\_id(0);

break;

}

}

}

else if(command=="add")

{

cin>>id>>re;

for(int i=0;i<10;i++)

{

if(booking[i].get\_id()==id)

{

if(!booking[i].reserveSeats(re))

cout<<"Can't perform this action.\n";

break;

}

}

}

else if(command=="cancel")

{

cin>>id>>re;

for(int i=0;i<10;i++)

{

if(booking[i].get\_id()==id)

{

if(!booking[i].cancelReservations(re))

cout<<"Can't perform this action.\n";

break;

}

}

}

else if(command!="quit")

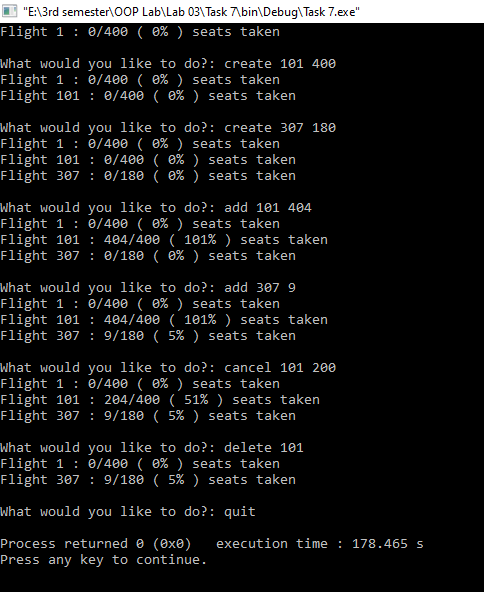
cout<<"Please enter a valid command.\n";

}

return 0;

}

**Output:**



# Activity # 08

**Title:**

GymSubsciption Class.

**In C++**

**Source code:**

#include <iostream>

#include<string>

using namespace std;

class GymSubcription

{

private:

int ID;

string name;

int membership;

public:

GymSubcription();

GymSubcription(int id, string Name, int Membership);

void set\_id(int num);

void set\_name(string Name);

int get\_id(){return ID;}

void ExtendMembership(int number\_of\_months);

void cancelMembership();

void printStatus();

};

GymSubcription::GymSubcription()

{

ID=0;name="";membership=0;

}

GymSubcription::GymSubcription(int id, string Name, int Membership)

{

ID=id;name=Name; membership=Membership;

}

void GymSubcription::set\_id(int num)

{

ID=num;

}

void GymSubcription::set\_name(string Name)

{

name=Name;

}

void GymSubcription::printStatus()

{

cout<<"Member "<<ID<<" : "<<name<<", subscription valid for "<<membership<<" months."<<endl;

}

void GymSubcription::ExtendMembership(int number\_of\_months)

{

membership+=number\_of\_months;

}

void GymSubcription::cancelMembership()

{

membership=0;

}

int main()

{

GymSubcription subcription[10];

int id,mem;

string Name;

string command="";

while(command!="quit")

{

for(int i=0;i<10;i++)

{

if(subcription[i].get\_id()!=0)

subcription[i].printStatus();

}

cout<<"\nWhat would you like to do?: ";

cin>>command;

if(command=="create")

{

cin>>id>>Name;

for (int i=0;i<10;i++)

{

if(subcription[i].get\_id()==0)

{

subcription[i]= GymSubcription(id,Name,0);

break;

}

}

}

else if(command=="delete")

{

cin>>id;

for(int i=0;i<10;i++)

{

if(subcription[i].get\_id()==id)

{

subcription[i].set\_id(0);

break;

}

}

}

else if(command=="extend")

{

cin>>id>>mem;

for(int i=0;i<10;i++)

{

if(subcription[i].get\_id()==id)

{

subcription[i].ExtendMembership(mem);

break;

}

}

}

else if(command=="cancel")

{

cin>>id;

for(int i=0;i<10;i++)

{

if(subcription[i].get\_id()==id)

{

subcription[i].cancelMembership();

break;

}

}

}

else if(command!="quit")

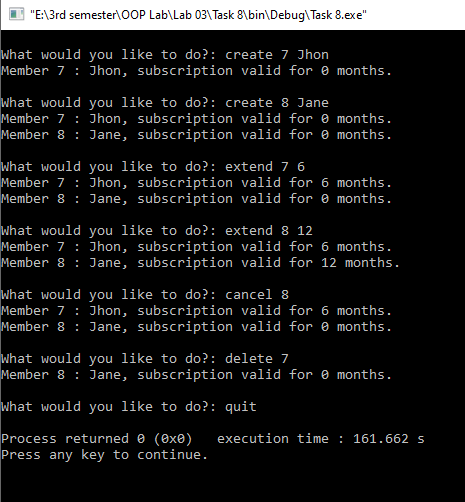
cout<<"Please enter a valid command.\n";

}

return 0;

}

**Output:**



# Activity # 09

**Title:**

Fraction Class.

**In C++**

**Source code:**

#include <iostream>

#include<string>

using namespace std;

class Fraction

{

private:

int numerator;

int denominator;

public:

Fraction(int numerator, int denominator);

string toString();

double toDouble();

};

Fraction::Fraction(int numerator,int denominator)

{

Fraction::numerator=numerator;

Fraction::denominator=denominator;

}

string Fraction::toString()

{

if(denominator<0)

{

numerator=-numerator;

denominator=-denominator;

}

int test=numerator;

numerator=numerator%denominator;

if(!numerator)

{

cout<<test/denominator;

return " ";

}

for (int i=-999999;;i++)

{

if (((denominator\*i)+numerator==test))

{

if(i!=0)

{

if (numerator<0)

numerator=-numerator;

cout<<i<<" "<<numerator<<"/"<<denominator;

return " ";

}

break;

}

}

cout<< numerator<<"/"<<denominator;

return " ";

}

double Fraction::toDouble()

{

return double(numerator)/double(denominator);

}

int main()

{

int num,den;

string input="";

getline(cin,input);

int i=0;

for (;input[i]!='/';i++);

string s=input.substr(0,i);

num=stoi(s);

s=input.substr(i+1);

den=stoi(s);

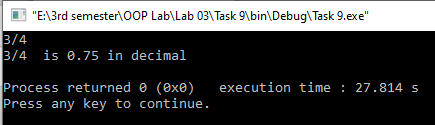
Fraction fraction(num,den);

cout<<fraction.toString()<<" is "<<fraction.toDouble()<<" in decimal"<<endl;

return 0;

}

**Output:**



# Activity # 10

**Title:**

Fraction:Part 2 Class.

**In C++**

**Source code:**

#include <iostream>

#include<string>

using namespace std;

class Fraction

{

private:

int numerator;

int denominator;

void reduce();

public:

Fraction(){numerator=0,denominator=0;}

Fraction(int numerator, int denominator);

string toString();

double toDouble();

Fraction plus(Fraction that);

Fraction minus(Fraction that);

Fraction times(Fraction that);

Fraction by(Fraction that);

};

void Fraction::reduce()

{

int a=numerator,b=denominator;

if (numerator<0)

a=-numerator;

if (denominator<0)

b=-denominator;

while (a!=b)

{

if(a>b)

a-=b;

if(b>a)

b-=a;

}

int gcd=a;

numerator/=gcd;

denominator/=gcd;

}

Fraction Fraction::times(Fraction that)

{

int num=this->numerator\*that.numerator;

int den=this->denominator\*that.denominator;

Fraction result(num,den);

result.reduce();

return result;

}

Fraction Fraction::plus(Fraction that)

{

int thisDen,thatDen;

int a=thisDen= this->denominator;

int b=thatDen= that.denominator;

int thisNum= this->numerator;

int thatNum= that.numerator;

while (a!=b)

{

if(a>b)

a-=b;

if(b>a)

b-=a;

}

int gcd=a;

int LCM=(this->denominator\*that.denominator)/gcd;

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

{

if(this->denominator\*i==LCM)

{

thisNum\*=i;

break;

}

}

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

{

if(that.denominator\*i==LCM)

{

thatNum\*=i;

break;

}

}

Fraction result((thisNum+thatNum),LCM);

result.reduce();

return result;

}

Fraction Fraction::minus(Fraction that)

{

int thisDen,thatDen;

int a=thisDen= this->denominator;

int b=thatDen= that.denominator;

int thisNum= this->numerator;

int thatNum= that.numerator;

while (a!=b)

{

if(a>b)

a-=b;

if(b>a)

b-=a;

}

int gcd=a;

int LCM=(this->denominator\*that.denominator)/gcd;

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

{

if(this->denominator\*i==LCM)

{

thisNum\*=i;

break;

}

}

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

{

if(that.denominator\*i==LCM)

{

thatNum\*=i;

break;

}

}

Fraction result(thisNum-thatNum,LCM);

result.reduce();

return result;

}

Fraction Fraction::by(Fraction that)

{

Fraction result;

result.numerator=this->numerator\*that.denominator;

result.denominator=this->denominator\*that.numerator;

result.reduce();

return result;

}

Fraction::Fraction(int numerator,int denominator)

{

this->numerator=numerator;

this->denominator=denominator;

}

string Fraction::toString()

{

if(denominator<0)

{

numerator=-numerator;

denominator=-denominator;

}

int test=numerator;

numerator=numerator%denominator;

if(!numerator)

{

cout<<test/denominator;

return " ";

}

for (int i=-999999;;i++)

{

if (((denominator\*i)+numerator==test))

{

if(i!=0)

{

if (numerator<0)

numerator=-numerator;

cout<<i<<" "<<numerator<<"/"<<denominator;

return " ";

}

break;

}

}

cout<< numerator<<"/"<<denominator;

return " ";

}

double Fraction::toDouble()

{

return double(numerator)/double(denominator);

}

int main()

{

Fraction fraction[2];

string input="";

for(int a=0;a<2;a++)

{

getline(cin,input);

int i=0;

for (;input[i]!='/';i++);

string s=input.substr(0,i);

int num=stoi(s);

s=input.substr(i+1);

int den=stoi(s);

fraction[a]= Fraction(num,den);

}

Fraction sum=fraction[0].plus(fraction[1]),sub=fraction[0].minus(fraction[1]);

Fraction prod=fraction[0].times(fraction[1]);

Fraction div=fraction[0].by(fraction[1]);

cout<<fraction[0].toString()<<" + ";

cout<<fraction[1].toString()<<" = ";

cout<<sum.toString()<<endl;

cout<<fraction[0].toString()<<" - ";

cout<<fraction[1].toString()<<" = ";

cout<<sub.toString()<<endl;

cout<<fraction[0].toString()<<" \* ";

cout<<fraction[1].toString()<<" = ";

cout<<prod.toString()<<endl;

cout<<fraction[0].toString()<<" / ";

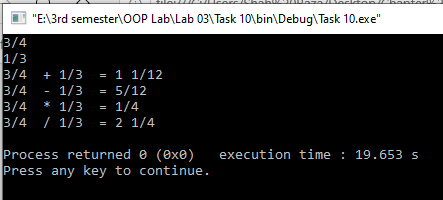
cout<<fraction[1].toString()<<" = ";

cout<<div.toString()<<endl;

return 0;

}

**Output:**



# Activity # 11

**Title:**

Fraction:Part 3 Class.

**In C++**

**Source code:**

#include <iostream>

#include<string>

using namespace std;

class Fraction

{

private:

int numerator;

int denominator;

void reduce();

public:

Fraction(){numerator=0,denominator=0;}

Fraction(int numerator, int denominator);

string toString();

double toDouble();

Fraction plus(Fraction that);

Fraction minus(Fraction that);

Fraction times(Fraction that);

Fraction by(Fraction that);

bool isGreaterThan(Fraction that);

bool isLessThan(Fraction that);

bool isEqual(Fraction that);

};

bool Fraction::isGreaterThan(Fraction that)

{

if((float(this->numerator)/float(this->denominator))>(float(that.numerator)/float(that.denominator)))

{

return true;

}

return false;

}

bool Fraction::isLessThan(Fraction that)

{

if((float(this->numerator)/float(this->denominator))<(float(that.numerator)/float(that.denominator)))

return true;

return false;

}

bool Fraction::isEqual(Fraction that)

{

if((float(this->numerator)/float(this->denominator))==(float(that.numerator)/float(that.denominator)))

return true;

return false;

}

void Fraction::reduce()

{

int a=numerator,b=denominator;

if (numerator<0)

a=-numerator;

if (denominator<0)

b=-denominator;

while (a!=b)

{

if(a>b)

a-=b;

if(b>a)

b-=a;

}

int gcd=a;

numerator/=gcd;

denominator/=gcd;

}

Fraction Fraction::times(Fraction that)

{

int num=this->numerator\*that.numerator;

int den=this->denominator\*that.denominator;

Fraction result(num,den);

result.reduce();

return result;

}

Fraction Fraction::plus(Fraction that)

{

int thisDen,thatDen;

int a=thisDen= this->denominator;

int b=thatDen= that.denominator;

int thisNum= this->numerator;

int thatNum= that.numerator;

while (a!=b)

{

if(a>b)

a-=b;

if(b>a)

b-=a;

}

int gcd=a;

int LCM=(this->denominator\*that.denominator)/gcd;

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

{

if(this->denominator\*i==LCM)

{

thisNum\*=i;

break;

}

}

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

{

if(that.denominator\*i==LCM)

{

thatNum\*=i;

break;

}

}

Fraction result((thisNum+thatNum),LCM);

result.reduce();

return result;

}

Fraction Fraction::minus(Fraction that)

{

int thisDen,thatDen;

int a=thisDen= this->denominator;

int b=thatDen= that.denominator;

int thisNum= this->numerator;

int thatNum= that.numerator;

while (a!=b)

{

if(a>b)

a-=b;

if(b>a)

b-=a;

}

int gcd=a;

int LCM=(this->denominator\*that.denominator)/gcd;

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

{

if(this->denominator\*i==LCM)

{

thisNum\*=i;

break;

}

}

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

{

if(that.denominator\*i==LCM)

{

thatNum\*=i;

break;

}

}

Fraction result(thisNum-thatNum,LCM);

result.reduce();

return result;

}

Fraction Fraction::by(Fraction that)

{

Fraction result;

result.numerator=this->numerator\*that.denominator;

result.denominator=this->denominator\*that.numerator;

result.reduce();

return result;

}

Fraction::Fraction(int numerator,int denominator)

{

this->numerator=numerator;

this->denominator=denominator;

}

string Fraction::toString()

{

if(denominator<0)

{

numerator=-numerator;

denominator=-denominator;

}

int test=numerator;

numerator=numerator%denominator;

if(!numerator)

{

cout<<test/denominator;

return " ";

}

for (int i=-999999;;i++)

{

if (((denominator\*i)+numerator==test))

{

if(i!=0)

{

if (numerator<0)

numerator=-numerator;

cout<<i<<" "<<numerator<<"/"<<denominator;

return " ";

}

break;

}

}

cout<< numerator<<"/"<<denominator;

return " ";

}

double Fraction::toDouble()

{

return double(numerator)/double(denominator);

}

int main()

{

Fraction fraction[2];

string input="";

for(int a=0;a<2;a++)

{

getline(cin,input);

int i=0;

for (;input[i]!='/';i++);

string s=input.substr(0,i);

int num=stoi(s);

s=input.substr(i+1);

int den=stoi(s);

fraction[a]= Fraction(num,den);

}

if (fraction[0].isGreaterThan(fraction[1]))

{

cout<<fraction[0].toString()<<" > ";

cout<<fraction[1].toString()<<endl;

}

if (fraction[0].isLessThan(fraction[1]))

{

cout<<fraction[0].toString()<<" < ";

cout<<fraction[1].toString()<<endl;

}

if (fraction[0].isEqual(fraction[1]))

{

cout<<fraction[0].toString()<<" = ";

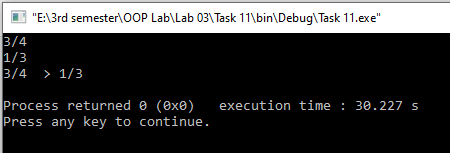
cout<<fraction[1].toString()<<endl;

}

return 0;

}

**Output:**



# Activity # 12

**Title:**

Point2D Class.

**In C++**

**Source code:**

#include <iostream>

#include <cmath>

using namespace std;

class Point2D

{

private:

double x;

double y;

public:

Point2D(){x=0,y=0;};

Point2D(double x, double y);

string toString();

double toDouble();

double distanceTo(Point2D that);

};

Point2D::Point2D(double x, double y)

{

this->x=x;

this->y=y;

}

double Point2D::distanceTo(Point2D that)

{

double distance=sqrt(pow((that.x-this->x),2)+pow((that.y-this->y),2));

return distance;

}

int main()

{

double x, y;

Point2D point[2];

for(int i=0;i<2;i++)

{

cin>>x>>y;

point[i]= Point2D(x,y);

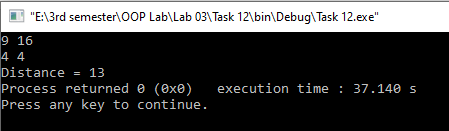
}

cout<<"Distance = "<<point[0].distanceTo(point[1]);

return 0;

}

**Output:**



# Activity # 13

**Title:**

Point2D:Part 2 Class.

**In C++**

**Source code:**

#include <iostream>

#include <cmath>

using namespace std;

class Point2D

{

private:

double x;

double y;

public:

Point2D(){x=0,y=0;};

Point2D(double x, double y);

string toString();

double toDouble();

double distanceTo(Point2D that);

double get\_x(){return x;}

double get\_y(){return y;}

};

class Line2D

{

private:

double slope;

double y\_intercept;

public:

Line2D(double slope,double y\_intercept);

Line2D(Point2D pointA, Point2D pointB);

string toString();

void print();

};

void Line2D::print()

{

cout<<"y = "<<slope<<" x ";

if(y\_intercept<0)

cout<<" - "<<y\_intercept<<endl;

else

cout<<" + "<<y\_intercept<<endl;

}

Line2D::Line2D(double slope, double y\_intercept)

{

this->slope=slope;

this->y\_intercept=y\_intercept;

}

Line2D::Line2D(Point2D pointA, Point2D pointB)

{

slope=((pointB.get\_y()-pointA.get\_y())/(pointB.get\_x()-pointA.get\_x()));

y\_intercept=((-slope)\*pointA.get\_x())+pointA.get\_y();

}

Point2D::Point2D(double x, double y)

{

this->x=x;

this->y=y;

}

double Point2D::distanceTo(Point2D that)

{

double distance=sqrt(pow((that.x-this->x),2)+pow((that.y-this->y),2));

return distance;

}

int main()

{

double x, y;

Point2D point[2];

for(int i=0;i<2;i++)

{

cin>>x>>y;

point[i]= Point2D(x,y);

}

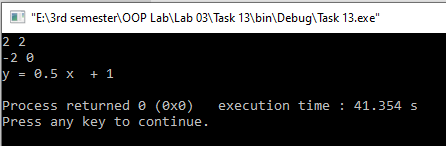
Line2D line(point[0],point[1]);

line.print();

return 0;

}

**Output:**



# Activity # 14

**Title:**

Point2D:Part 3 Class.

**In C++**

**Source code:**

#include <iostream>

#include <cmath>

using namespace std;

class Point2D

{

private:

double x;

double y;

public:

Point2D(){x=0,y=0;};

Point2D(double x, double y);

string toString();

double toDouble();

double distanceTo(Point2D that);

double get\_x(){return x;}

double get\_y(){return y;}

};

class Line2D

{

private:

double slope;

double y\_intercept;

public:

Line2D(double slope,double y\_intercept);

Line2D(Point2D pointA, Point2D pointB);

string toString();

void print();

bool contains(Point2D point);

};

bool Line2D::contains(Point2D point)

{

if(point.get\_y()==(slope\*point.get\_x())+y\_intercept)

return true;

else

return false;

}

void Line2D::print()

{

cout<<"y = "<<slope<<" x ";

if(y\_intercept<0)

cout<<" - "<<y\_intercept<<endl;

else

cout<<" + "<<y\_intercept<<endl;

}

Line2D::Line2D(double slope, double y\_intercept)

{

this->slope=slope;

this->y\_intercept=y\_intercept;

}

Line2D::Line2D(Point2D pointA, Point2D pointB)

{

slope=((pointB.get\_y()-pointA.get\_y())/(pointB.get\_x()-pointA.get\_x()));

y\_intercept=((-slope)\*pointA.get\_x())+pointA.get\_y();

}

Point2D::Point2D(double x, double y)

{

this->x=x;

this->y=y;

}

double Point2D::distanceTo(Point2D that)

{

double distance=sqrt(pow((that.x-this->x),2)+pow((that.y-this->y),2));

return distance;

}

int main()

{

double x, y;

Point2D point[3];

for(int i=0;i<3;i++)

{

cin>>x>>y;

point[i]= Point2D(x,y);

}

Line2D line(point[0],point[1]);

if(line.contains(point[2]))

cout<<"Collinear.\n";

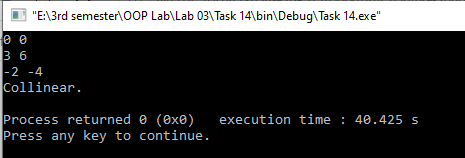
else

cout<<"Non-Collinear.\n";

return 0;

}

**Output:**



# Activity # 15

**Title:**

FarmAnimal Class.

**In C++**

**Source code:**

#include <iostream>

using namespace std;

class FarmAnimal

{

private:

double water\_consumption;

public:

FarmAnimal(double water\_consumption);

double getWaterConsumption();

};

FarmAnimal::FarmAnimal(double water\_consumption)

{

this->water\_consumption=water\_consumption;

}

double FarmAnimal::getWaterConsumption()

{

return water\_consumption;

}

class DummyAnimal : public FarmAnimal

{

public:

DummyAnimal();

};

DummyAnimal::DummyAnimal(): FarmAnimal(10.0)

{

}

class DoublingAnimal: public FarmAnimal

{

public:

DoublingAnimal(double given\_water\_consumption);

};

DoublingAnimal::DoublingAnimal(double given\_water\_consumption): FarmAnimal(2\*given\_water\_consumption)

{

}

void printConsumption(FarmAnimal animal)

{

cout<<"This Animal Consumes "<< animal.getWaterConsumption()<<" liters of water\n";

}

int main()

{

FarmAnimal animalA(5);

DummyAnimal animalB;

DoublingAnimal animalC(21);

cout << "animalA consumes " << animalA.getWaterConsumption()<< " liters of water." << endl;

cout << "What about the others?" << endl;

printConsumption(animalB);

printConsumption(animalC);

return 0;

}

**Output:**

