**1. Consider the following class C1**:

class C1

{

void add(int p,int q)

{

System.out.println(p+q);

}

}

There is another class C2 which extends class C1:

class C2 extends C1

{

void mul(int p,int q)

{

System.out.println(p\*q);

}

}

So we can make of a object of class C2 which can use both mul and add methods.

Main function is already created in the editor and instance of C2 is also made.

So your task is to add a method in class C2 naming task() which will take 2 parameters as input (a and b) and print the sum of their squares i.e a2+b2. Changes in the main function has been made already.

Input

Two integers a and b.

Output

Print three lines. In the first line, print the sum a and b. In the second line, print the product of a and b. In the third line, print the sum of the squares a and b.

Sample Input

2 4

Sample Output

6

8

20

**2. Suppose that the following two classes have been declared**:

public class Car {

public void m1() {

System.out.println("car 1");

}

​

public void m2() {

System.out.println("car 2");

}

​

public String toString() {

return "vroom";

}

}

public class Truck extends Car {

public void m1() {

System.out.println("truck 1");

}

public void m2() {

super.m1();

}

public String toString() {

return super.toString() + super.toString();

}

}

Copy these classes.

Write a class MonsterTruck which inherits Truck whose methods have the behavior below.

Method Output/Return

m1 monster 1

m2 truck 1

car 1

toString "monster vroomvroom"

In the main method, create an instance of MonsterTruck and output the return values of m1(), m2() and toString().

Hint:- Pass the MonsterTruck object in System.out.println() to print the return value of toString().

**3. Consider the following Sports class**:

class Sports{

String getName(){

return "Generic Sports";

}

void getNumberOfTeamMembers(){

System.out.println( "Each team has n players in " + getName() );

}

}

Next, we create a Soccer class that inherits from the Sports class. We can override the getName method and return a different, subclass-specific string:

class Soccer extends Sports{

@Override

String getName(){

return "Soccer Class";

}

}

Note: When overriding a method, you should precede it with the @Override annotation. The parameter(s) and return type of an overridden method must be exactly the same as those of the method inherited from the supertype.

**Task**

Override the getNumberOfTeamMembers() method in the Soccer class so that it replaces n with 11.

In the main() method, create instances of Sports and Soccer and use their respective getName() and getNumberOfTeamMembers() methods to output the following:

Generic Sports

Each team has n players in Generic Sports

Soccer Class

Each team has 11 players in Soccer Class

**4. Following is an example of abstract class:**

abstract class Book{

String title;

abstract void setTitle(String s);

String getTitle(){

return title;

}

}

You have to create another class that extends the abstract class. Then you can create an instance of the new class.

Notice that setTitle method is abstract too and has no body. That means you must implement the body of that method in the child class.

Write a class called MyBook that extends the abstract class and implements the setTitle() method. In main(), create an instance of MyClass and use the setTitle() method to set a title. Afterwards, print the title.

Sample Input

A tale of two cities

Sample Output

The title is: A tale of two cities