Write a class for instantiating the objects that represent the two-dimensional Cartesian coordinate system.

A. make a particular member function of one class to friend function in another class for addition

B. make other three functions to work as a bridge between the classes for multiplication, division and subtraction.

C. Also write a small program to demonstrate that all the member functions of one class are the friend functions of another class if the former class is made friend to the latter.

Make least possible classes to demonstrate all above in single program without conflict.

```
#include <iostream>
#include <cmath>
#define SUCCESS 0
using namespace std;
class Coordinate;
class Vector;
class Polar
{
private:
  float radius , theta;
public:
  Polar (float r ,float angle ):radius(r),theta(angle){};
  friend class Vector; // let Vector acess all its private data types
  Coordinate toCartesian(); // not defined here due to incomplete type error
};
class Vector
{
public:
  // not defined here due to incomplete type error

  Coordinate add(Coordinate a, Coordinate b);
  Coordinate sub(Coordinate a, Coordinate b);
  Coordinate mul(Coordinate a, Coordinate b);
  Coordinate div(Coordinate a, Coordinate b);
```

```
};

class Coordinate
{
private:
  float x, y;
public:
  Coordinate(float a, float b):x(a),y(b){};
  friend Coordinate Vector::add(Coordinate a, Coordinate b); // let Vectors add
member access private members
  friend Coordinate Vector::sub(Coordinate a, Coordinate b); // let Vecors sub member
access private members
  Polar toPolar()
  {
    Polar temp(sqrt(x*x+y*y),atanf(y/x));
    return temp;
  }
  void display()
  {
    cout << "(" << x << "," << y <<  ")";
  }
};

Coordinate Polar::toCartesian()
  {
    Coordinate C(radius*cos(theta), radius *sin(theta));
    return C;
  }
Coordinate Vector::add(Coordinate a, Coordinate b)
  {
    Coordinate temp(a.x+b.x, a.y+b.y);
    return temp;
  }
Coordinate Vector::sub(Coordinate a, Coordinate b)
  {
    Coordinate temp(a.x-b.x, a.y-b.y);
    return temp;
  }
Coordinate Vector::mul(Coordinate a, Coordinate b)
  {
```

```
  Polar pa = a.toPolar();
  Polar pb = b.toPolar();
  Polar p(pa.radius*pb.radius,pa.theta+pb.theta);
  return p.toCartesian();

 }
Coordinate Vector::div(Coordinate a, Coordinate b)
 {
  Polar pa = a.toPolar();
  Polar pb = b.toPolar();
  Polar p(pa.radius/pb.radius,pa.theta-pb.theta);
  return p.toCartesian();
 }

int main()
{
 int x , y;
 char temp; // garbage value of ,
 cout << "Enter cordinate x y in format x,y";
 cin >> x >> temp >> y;
 Coordinate a(x,y);
 cout << "Enter cordinate x y in format x,y";
 cin >> x >> temp >> y;
 Coordinate b(x,y);
 Vector v;
 Coordinate c = v.add(a,b);
 cout << "The sum is ";
 c.display();
 cout << endl;
 Coordinate d = v.sub(a,b);
 cout << "The difference is";
 d.display();
 cout << endl;
 Coordinate p = v.mul(a,b);
 cout << "The product is";
 p.display();
 cout << endl;
 Coordinate q = v.div(a,b);
 cout << "The quotient is";
 q.display();
```

```
  cout << endl;
  return SUCCESS;
}

#include<iostream>//a.or
using namespace std;
class class2;
class class1
{
   int x,y;
public:
   class1 (int ix,int iy)
   {
      x=ix;
      y=iy;
   }
   friend void add(class1 c1, class2 c2);
   friend void multiply (class1 c1, class2 c2);
   friend void division (class1 c1, class2 c2);
   friend void subtraction (class1 c1, class2 c2);
};
class class2
{
    int x,y;
public:
   class2 (int ix,int iy)
   {
      x=ix;
      y=iy;
   }
   friend void add(class1 c1, class2 c2);
   friend void multiply (class1 c1, class2 c2);
   friend void division (class1 c1, class2 c2);
   friend void subtraction (class1 c1, class2 c2);
};
void add(class1 c1, class2 c2)
{
   cout<<"Sum of given
coordinates:"<<endl<<"x="<<c1.x+c2.x<<endl<<"y="<<c1.y+c2.y<<endl;
}
```

```
void multiply(class1 c1, class2 c2)
{
   cout<<"Product of given
coordinates:"<<endl<<"x="<<c1.x*c2.x<<endl<<"y="<<c1.y*c2.y<<endl;
}
void division (class1 c1, class2 c2)
{
   cout<<"Division of given
coordinates:"<<endl<<"x="<<static_cast<float>(c1.x)/c2.x<<endl<<"y="<<static_cast<f
loat>(c1.y)/c2.y<<endl;
}
void subtraction (class1 c1, class2 c2)
{
   cout<<"Difference of given coordinates:"<<endl<<"x="<<c1.x-
c2.x<<endl<<"y="<<c1.y-c2.y<<endl;
}
int main()
{
   class1 c1(3,5);
   class2 c2(6,8);
   add(c1,c2);
   multiply(c1,c2);
   division(c1,c2);
   subtraction(c1,c2);
}

#include<iostream>//b.or
using namespace std;
class class2;
class class1
{
   int x,y;
public:
   class1 (int ix,int iy)
   {
      x=ix;
      y=iy;
   }
friend class2;
};
```

```
class class2
{
     int x,y;
public:
   class2 (int ix,int iy)
   {
      x=ix;
      y=iy;
   }
   void add(class1 c1, class2 c2)
   {
      cout<<"Sum of given
coordinates:"<<endl<<"x="<<c1.x+c2.x<<endl<<"y="<<c1.y+c2.y<<endl;
   }
   void multiply (class1 c1, class2 c2)
   {
      cout<<"Product of given
coordinates:"<<endl<<"x="<<c1.x*c2.x<<endl<<"y="<<c1.y*c2.y<<endl;
   }
   void division (class1 c1, class2 c2)
   {
      cout<<"Division of given
coordinates:"<<endl<<"x="<<static_cast<float>(c1.x)/c2.x<<endl<<"y="<<static_cast<f
loat>(c1.y)/c2.y<<endl;
   }
   void subtraction (class1 c1, class2 c2)
   {
      cout<<"Difference of given coordinates:"<<endl<<"x="<<c1.x-
c2.x<<endl<<"y="<<c1.y-c2.y<<endl;
   }
};
int main()
{
   class1 c1(3,5);
   class2 c2(6,8);
   c2.add(c1,c2);
   c2.multiply(c1,c2);
   c2.division(c1,c2);
   c2.subtraction(c1,c2);
}
```