```
// Fichier Point.h

#ifndef __POINT__
#define __POINT__

class Point {
  private :
    float my_abs, my_ord;
    static const float epsilon;
  public :
    Point( float x, float y );                  // constructeur
    Point();                                    // constructeur par defaut
    ~Point();                                   // destructeur
    Point( const Point & p );                   // constructeur par copie
    Point & operator=( const Point & p );       // operateur d'affectation
    void affiche() const;
    float getX() const;
    float getY() const;
    void setX( float new_x );
    void setY( float new_y );
    // quelques services
    void deplace( float d_x, float d_y );
    float distance() const;
    float distance( const Point & p ) const;
    bool operator==( const Point & p ) const;
    bool operator<( const Point & p) const;
};

#endif
```



```
// Fichier Point.cc

#include <iostream>
#include <cmath>
#include "Point.h"

using namespace std;

const float Point::epsilon=0.00000001;

Point::Point( float x, float y )
{
  cout << "Point::Constructeur : " << x << " , " << y << endl;
  my_abs = x;
  my_ord = y;
}

Point::Point()
{
  cout << "Point::Constructeur par defaut" << endl;
  my_abs = 0;
  my_ord = 0;
}

Point::~Point()
{
  cout << "Point::Destructeur" << endl;
}

Point::Point( const Point & p )
{
  cout << "Point::Constructeur par copie" << endl;
  my_abs = p.my_abs;
  my_ord = p.my_ord;
}

Point &
Point::operator=( const Point & p )
{
  cout << "Point::Operateur affectation" << endl;
  if ( this != &p ) {
    my_abs = p.my_abs;
    my_ord = p.my_ord;
  }
  return *this;
}

void
Point::affiche() const
{
  cout << "(" << my_abs << "," << my_ord << ")" << endl;
}

float
Point::getX() const
{
  return my_abs;
}

float
Point::getY() const
{
  return my_ord;
}

void
Point::setX( float new_x )
```

```
{
  my_abs = new_x;
}

void
Point::setY( float new_y )
{
  my_ord = new_y;
}

void
Point::deplace( float d_x, float d_y )
{
  my_abs += d_x;
  my_ord += d_y;
}

float
Point::distance() const
{
  return sqrt(my_abs*my_abs + my_ord*my_ord);
}

float
Point::distance( const Point & p ) const
{
  float carrex = (p.my_abs-my_abs) * (p.my_abs-my_abs);
  float carrey = (p.my_ord-my_ord) * (p.my_ord-my_ord);
  return sqrt(carrex + carrey);
}

bool
Point::operator==( const Point & p ) const
{
   return distance(p) < epsilon ;
}

bool
Point::operator<( const Point & p) const
{
  bool rep = false ;
  // si points egaux : retourner faux
  if ( distance( p ) >= epsilon ) {
    // points pas egaux
    if ( my_abs < p.my_abs )
      rep = true ;
    else if ( my_abs > p.my_abs )
      rep = false ;
    else {  // abscisses egales
      if ( my_ord < p.my_ord )
        rep = true ;
      else
        rep = false ;
    }
  }
  return rep ;
}
```



```
// Fichier main.cc
#include <iostream>
#include "Point.h"

using namespace std;

Point
milieu( const Point & a, const Point & b ) {
  Point m;
  m.setX( (a.getX()+b.getX())/2 );
  m.setY( (a.getY()+b.getY())/2 );
  return m;
}

int main()
{
  cout << endl << "Jeu de tests de la classe Point" << endl << endl ;

  Point p1( 4, 5 );
  Point p2( 3.5, 8.2 );

  p1.affiche() ;
  p2.affiche() ;

  p2.deplace( 6, -1.2 );
  cout << "apres deplacement (6, -1.2) :" << endl;
  cout << p2.getX() << "," << p2.getY() << endl;
  p2.affiche();
  cout << "distance origine : " << p2.distance() << endl;

  Point p;
  p.affiche();

  cout << "Point au milieu : " << endl ;
  milieu(p1, p2).affiche();

  Point u( 4, 4 );
  Point v( 4, 4.0001 );

  cout << "distance : " << u.distance( v ) << endl ;

   if ( u.operator==( v ) )
    cout << "operator==: egal" << endl ;
  else
    cout << "operator==: pas egal" << endl ;

  if ( u==v )
    cout << "==: egal" << endl ;
  else
  cout << "==: pas egal" << endl ;

  if ( u<v )
    cout << "operator<: strictement inferieur" << endl ;
  else
    cout << "operator<: pas strictement inferieur" << endl ;

  Point x;
  x = u;
  x.affiche();
  u.affiche();

  return 0;
}


/*
Jeu de tests de la classe Point
```

```
Point::Constructeur : 4 , 5
Point::Constructeur : 3.5 , 8.2
(4,5)
(3.5,8.2)
apres deplacement (6, −1.2) :
9.5 , 7
(9.5,7)
distance origine : 11.8004
Point::Constructeur par defaut
(0,0)
Point au milieu :
Point::Constructeur par defaut
(6.75,6)
Point::Destructeur
Point::Constructeur : 4 , 4
Point::Constructeur : 4 , 4.0001
distance : 0.000100136
operator==: pas egal
==: pas egal
operator<: strictement inferieur
Point::Constructeur par defaut
Point::Operateur affectation
(4,4)
(4,4)
Point::Destructeur
Point::Destructeur
Point::Destructeur
Point::Destructeur
Point::Destructeur
Point::Destructeur

*/
```