# Exercice 1: Questions de cours (1+0.5x5+1+1+1=6 points)

1. Les extensions d’un fichier source C++ sont : .cpp et .h.
2. Definitions des concepts:
   * **Programmation orientée objet** c’est un paradigme de programmation inventé au début des années 1960 consitant en la définition et l’interaction de briques logicielles appelées objets.
   * **Un objet est une structure informatique regroupant** :
     + des variables, caractérisant l’état de l’objet,
     + des fonctions, caractérisant le comportement de l’objet.
   * **Classe**: Une classe est un ensemble d’ojets de même type.
   * **Polymorphisme**: c’est la capacité d’un l’objet à posséder plusieurs formes.
   * **Héritage**: C’est un mécanisme de la programmation orientée objet qui permet de créer des classes dites filles à partir des caractéristique et méthodes des classes existantes dites parentes.
3. Signification des mots clés:
   * **Virtual**: signifie que toute fonction-membre de la classe de base doit être surchargée (c’est-à-dire redéfinie) dans une classe dérivée.
   * **Private**: signifie que les propriétés et méthodes d’une classe sont privées à cette classe et inaccéssible depuis l’exterieur.
4. La sortie du code est: **15**.

# Solution de l’exercice 2 (choisir la(les) bonne(s) reponse(s)):(0.25x8=2pts)

1. c)une instance / la classe
2. b) type dynamique / type statique
3. b)Square/Shape
4. a)\*p = a;
5. a)Avant
6. b)après
7. peut contenir au moins constructeurs—Il n’est pas nécessaire de définir explicitement un constructeur ; cependant, toutes les classes doivent avoir au moins un constructeur. Un constructeur vide par défaut sera généré si vous n’en fournissez pas .
8. d)du nombre ou du type de leurs paramètres.

# Exercice 1

Réalisation d’une classe point permettant de manipuler un point d’un plan.:

1. Ajoutez les propriétes privées de cette classe sachant que un point est défini par ses coordonnées x et y (des membres privés)

* private:  
   // Coordinates of the point.  
   double x;  
   double y;

1. Ajoutez un constructeur par défault et un constructeur paramétré.

* public:  
   // Constructors  
   Point(); // Default constructor  
   Point(double x, double y); // Two-argument constructor

1. Ajoutez une fonction membre déplace effectuant une translation définie par ses deux arguments dx et dy (double)

* point point::deplace(double dx, double dy)  
  {  
    
   set\_x(get\_x() + dx);  
    
   set\_y(get\_y() + dy);  
   return \*this;  
  }

1. Ajoutez une fonction membre affiche se contentant d’afficher les coordonnées cartésiennes du point.

* void point::affiche()  
  {  
   cout << "les coordonnees sont:" << endl;  
   cout << "x=" << get\_x() << endl;  
   cout << "y=" << get\_y() << endl;  
  }

1. Ajoutez une fonction membre saisir se contentant de saisir les coordonnées cartésiennes du point.

* void point::saisir()  
  {  
   cout << "donnee les coordonnees:" << endl;  
   cout << "x=" << endl;  
   cin >> x;  
   cout << "y=" << endl;  
   cin >> y;  
  }

1. Ajoutez une fonction membre distance effectuant calculant la distance entre deux point.

* double point::distance(point &p)  
  {  
   double p1, x1, x2;  
   x1 = (get\_x() - p.get\_x()) \* (get\_x() - p.get\_x());  
   x2 = (get\_y() - p.get\_y()) \* (get\_y() - p.get\_y());  
   //p1=sqrt(((get\_x()-p.x)\*((get\_x()-p.x))+((get\_y()-p.y)\*(get\_y()-p.y)));  
   p1 = sqrt(x1 + x2);  
   return p1;  
  }

1. Ajoutez une une fonction membre milieu donnant le milieu d’un segment.

* point point::milieu(point &p)  
  {  
   point p1;  
   p1.x = (get\_x() + p.get\_x()) / 2;  
   p1.y = (get\_y() + p.get\_y()) / 2;  
   return p1;  
  }

1. Ecrivez un petit programme d’essai (main) gérant la classe point.

* #include <iostream>  
  #include "point.h"  
  void main()  
  {  
   point p(1, 1);  
   point x(5, 5);  
   point c;  
    
   p.affiche();  
   p.deplace(5, 5);  
   p.affiche();  
   cout << "la distance est : " << p.distance(x);  
   c = p.milieu(x);  
   c.affiche();  
  }