Série 5 : Programmation C++ (Les classes-Heritage)

Exercice 1 :

Quels seront les résultats fournis par l'exécution du programme suivant (ici, la déclaration de la classe `demo`, sa définition et le programme d'utilisation ont été regroupés en un seul fichier) :

```
#include <iostream>
using namespace std ;
class demo
{  int x, y ;
  public :
   demo (int abs=1, int ord=0)    // constructeur I (0, 1 ou 2 arguments)
     { x = abs ;  y = ord ;
       cout << "constructeur I            : " << x << " " << y << "\n" ;
     }
   demo (demo &) ;                // constructeur II (par recopie)
   ~demo () ;                     // destructeur
} ;
demo::demo (demo & d)            // ou demo::demo (const demo & d)
{  cout << "constructeur II (recopie) : " << d.x << " " << d.y << "\n" ;
   x = d.x ;  y = d.y ;
}
demo::~demo ()
{  cout << "destruction               : " << x << " " << y << "\n" ;
}
main ()
{  void fct (demo, demo *) ;         // proto fonction indépendante fct
   cout << "début main\n" ;
   demo a ;
   demo b = 2 ;
   demo c = a ;
   demo * adr = new demo (3,3) ;
   fct (a, adr) ;
   demo d = demo (4,4) ;
   c = demo (5,5) ;
   cout << "fin main\n" ;
}
void fct (demo d, demo * add)
{   cout << "entrée fct\n" ;
    delete add ;
    cout << "sortie fct\n" ;
}
```

Exercice 2 :

Créer une classe `point` ne contenant qu'un constructeur sans arguments, un destructeur et un membre donnée privé représentant un numéro de point (le premier créé portera le numéro 1, le suivant le numéro 2...). Le constructeur affichera le numéro du point créé et le destructeur affichera le numéro du point détruit. Écrire un petit programme d'utilisation créant dynamiquement un tableau de 4 points et le détruisant.

Exercice 3 :

1. Réaliser une classe nommée `set_int` permettant de manipuler des ensembles de nombres entiers. On devra pouvoir réaliser sur un tel ensemble les opérations classiques suivantes : lui ajouter un nouvel élément, connaître son cardinal (nombre d'éléments), savoir si un entier donné lui appartient.
Ici, on conservera les différents éléments de l'ensemble dans un tableau alloué dynamiquement par le constructeur. Un argument (auquel on pourra prévoir une valeur par défaut) lui précisera le nombre maximal d'éléments de l'ensemble.
2. Écrire, en outre, un programme (`main`) utilisant la classe `set_int` pour déterminer le nombre d'entiers différents contenus dans 20 entiers lus en données.
3. Que faudrait-il faire pour qu'un objet du type `set_int` puisse être transmis par valeur, soit comme argument d'appel, soit comme valeur de retour d'une fonction ?

Exercice 4 :

On dispose d'un fichier `point.h` contenant la déclaration suivante de la classe `point` :

```
#include <iostream>
using namespace std ;
class point
{   float x, y ;
  public :
    point (float abs=0.0, float ord=0.0)
       { x = abs ; y = ord ;
       }
    void affiche ()
       { cout << "Coordonnées : " << x << " " << y << "\n" ;
       }
    void deplace (float dx, float dy)
       { x =  x + dx ;  y = y + dy ;
       }
} ;
```

a. Créer une classe `pointcol`, dérivée de `point`, comportant :

• un membre donnée supplémentaire `cl`, de type `int`, contenant la « couleur » d'un point ;

• les fonctions membre suivantes :

`affiche (redéfinie), qui affiche les coordonnées et la couleur d'un objet de type pointcol ;`

`colore (int couleur), qui permet de définir la couleur d'un objet de type pointcol,`

un constructeur permettant de définir la couleur et les coordonnées (on ne le définira pas en ligne).

b. Que fera alors précisément cette instruction :

```
   pointcol (2.5, 3.25, 5) ;
```