ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communications
Cours de Programmation Orientée Objet, sections MA et PH
Chappelier J.-C.

NOM : Hanon Ymous
(000000)
Place : 0

#0000

Programmation Orientée Objet (SMA/SPH) : Examen final

2 juin 2022

INSTRUCTIONS (à lire attentivement)

IMPORTANT ! Veuillez suivre les instructions suivantes à la lettre sous peine de voir votre examen annulé dans le cas contraire.

1. Vous disposez d'une heure quarante-cinq minutes pour faire cet examen (9h15 – 11h00).
2. Vous devez **écrire à l'encre noire ou bleu foncée**, pas de crayon ni d'autre couleur.
   N'utilisez **pas non plus de stylo effaçable** (perte de l'information à la chaleur).
3. Vous avez droit à toute documentation papier.
   En revanche, vous ne pouvez pas utiliser d'ordinateur personnel, ni de téléphone portable, ni aucun autre matériel électronique.
4. Répondez aux questions directement sur la donnée ; ne joignez aucune feuille supplémentaire ; **seul ce document sera corrigé**.
   Vous disposez, si nécessaire, d'une page blanche supplémentaire en fin de sujet.
5. Lisez attentivement et *complètement* les questions de façon à ne faire que ce qui vous est demandé. Si l'énoncé ne vous paraît pas clair, ou si vous avez un doute, demandez des précisions à l'un des assistants.
6. L'examen comporte quatre exercices indépendants (sur 16 pages), qui peuvent être traités dans n'importe quel ordre, mais qui ne rapportent pas la même chose (les points sont indiqués, le total est de 115 points) ; tous les exercices comptent pour la note finale :
   — questions « courtes » : 24 points ;
   — conception : 35 points ;
   — trouver les erreurs : 26 points ; et
   — déroulement de programmes : 30 points.

Question 1 – Questions de cours [24 points]

1.1 La classe ! [2 points]

Considérez le code ci-dessous :

```
1   class C {
2    public:
3     C() {}
4     C(double a = 7.1) : c(a) {}
5    protected:
6     double c;
7   };
8
9   int main() {
10    C mon_c;
11    return 0;
12  }
```

Entourer la ou les option(s) ci-dessous qui sont correcte(s). (pénalités pour réponses fausses entourées.)

1. Le code ne compile pas.
2. Le code compile, mais produit un erreur si on l'exécute.
3. Le code compile et, à la ligne 10, le constructeur `C::C(double)` est appelé.
4. Le code compile et, à la ligne 10, le constructeur par défaut est appelé.

Justifiez brièvement votre réponse :

1.2 Ayants droit [5 points]

Considérez une classe C++. Dans quelles conditions décrites ci-dessous une méthode/fonction peut elle accéder aux méthodes privées/protégées/publiques de cette classe ? Mettez un plus (+) si l'accès est possible et un moins (-) si c'est interdit. Laissez blanc si vous ne savez pas (pénalité pour réponse fausse).

	peut accéder à une méthode :		
	`private`	`protected`	`public`
un opérateur surchargé en externe			
un opérateur surchargé en interne			
une méthode de la même classe			
une méthode d'une autre classe quelconque			
fonction quelconque			
une fonction `friend` de la classe			
une méthode d'une sous-classe, accès direct			
une méthode d'une sous-classe, accès au travers d'un argument			

1.3 Les losanges sont éternels [7 points]

Considérez les classes ci-dessous :

```
class X {
public:
  virtual double g() = 0;
};

class Y : public X {
public:
  double g() { ... }
};

class Z : public X {
 public:
  double g() { ... }
};
```

Quel problème a-t-on si l'on ajoute une quatrième classe, disons `A`, qui hérite de `Y` *et* de `Z` ?

Proposez des solutions *sans* modifier le code founi, et discutez leurs avantages et inconvénients.

Est-ce que si `X` était une classe *virtuelle* cela changerait votre réponse ? Si oui, comment/en quoi ?

suite au dos ☞

1.4 Concepts [10 points]

Expliquez clairement en *une* phrase les concepts suivants et donnez *un* exemple illustratif *simple* mais *pertinent* en code C++ :

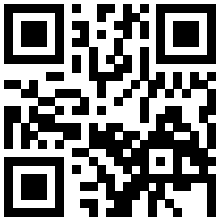
1. classe virtuelle

2. attribut de classe

3. méthode constante

4. méthode virtuelle pure

5. `this`



Question 2 Conception OO et programmation [35 points]

Un fleuriste souhaite pouvoir gérer la composition des bouquets qu'il peut proposer à sa clientèle.

Les bouquets sont composés de fleurs et de décorations (p.ex. tiges, branches, feuilles, tissus, ...). On parle ici de prestation (bouquets) et de « modèles de fleurs présentes en catalogue », pas des vrais bouquets effectivement faits avec les vraies fleurs. Par exemple, la « fleur » « `tulipe` » désigne ci-dessous un type de fleurs présent dans le catalogue du fleuriste et non pas une tulipe concrète.

Tous les composants d'un bouquet sont caractérisés par :
- leur nom (une chaîne de caractères, unique pour chaque composant) ;
- leur couleur ;
- leur prix à l'unité (un `double`).

Les décoration ont en plus une rigidité (disons : soit souple, soit solide) et les fleurs une odeur (disons : parmi 8 odeurs codifiées).

Le fleuriste souhaite gérer son stock de composants (fleur ou décoration) comme un ensemble de paires (composant, quantité en stock) ; p.ex. : `(rose, 45), (tige de bambou, 23), (tulipes, 85)`.

Par ailleurs, un bouquet est caractérisé par :
- un nom (une chaîne de caractères, unique pour chaque bouquet) ;
- un ensemble de paires (composant, quantité de ce composant présente dans le bouquet) ;
  p.ex. : `(tulipe, 5), (rose, 3), (tige de bambou, 1)` ;
  pour simplifier, on supposera que tous les composants possibles existent plus longtemps que tous les bouquets (les composants sont présents en catalogue plus longtemps que les bouquets ; ils restent présents en catalogue même si la quantité en stock est 0) ;
- une marge (`double`), représentant le prix à ajouter à la somme des prix de base des composants pour avoir le prix total du bouquet.

Le fleuriste souhaite se doter d'un programme permettant au moins :

1. de débiter/augmenter d'une quantité donnée le stock d'un composant donné (par exemple : augmenter de 150 le nombre de tulipes en stock) ;
2. de modifier la quantité d'un composant donné (fleur ou décoration) dans un bouquet donné ;
3. d'afficher l'état détaillé du stock (noms et quantités disponibles) ;
4. de calculer (pas afficher !) le prix d'un bouquet (somme des prix des composants plus la marge) ;
5. de tester si les quantités en stock sont suffisantes pour un nombre donné de bouquets (identiques) ; par exemple de tester si l'on peut faire 7 bouquets qui s'appellent « réveil enchanté » ;
6. de composer un bouquet donné : cette fonction doit tester si les stock sont suffisants pour faire le bouquet, et, si oui, elle doit débiter le stock du nombre de composants nécessaire à ce bouquet.

   Par exemple si la fonction compose un bouquet constitué de 5 tulipes, 3 roses et une tige de bambou, il doit diminuer le stock de 5 tulipes, 3 roses et une tige de bambou et retourner `true`, ou afficher un message et retourner `false` si le stock est insuffisant pour composer le bouquet.

2.1 Conception (22 points)

Sans donner tout le détail du code complet (on ne demande ici qu'une *conception*), écrivez **en C++** les classes, les éventuelles relations d'héritage, les attributs et les méthodes des classes, les droits d'accès et les éventuelles fonctions (externes) que vous utiliseriez pour implémenter un tel programme.

Précisez les types des attributs et les prototypes des méthodes/fonctions, mais ne donnez pas leur définition (on répète : il s'agit ici de la partie *conception*, pas de l'implémentation ; c.-à-d. les prototypes, pas les définitions des méthodes).

Lorsque c'est nécessaire, indiquez également les constructeurs et les destructeurs (sans leur définition).

Veuille en particulier donner explicitement les *prototypes*[1] des fonctions/méthodes nécessaires aux six fonctionnalités désirées par le fleuriste.
Dites à chaque fois clairement si c'est une fonction ou une méthode (et de quelle classe).

1. **pas** les définitions

Suite de la réponse :

suite au dos ☞

2.2 Programmation (13 points)

Écrire (c.-à-d. donner les définitions) des méthodes/fonctions demandées par le fleuriste aux points 2, 4 et 6 de la page 5 :

2. modifier la quantité d'un composant donné (fleur ou décoration) dans un bouquet donné ;
4. calculer (pas afficher !) le prix d'un bouquet (somme des prix des composants plus la marge) ;
6. composer un bouquet donné : cette fonction doit tester si les stock sont suffisants pour faire le bouquet, et, si oui, elle doit débiter le stock du nombre de composants nécessaire à ce bouquet. Par exemple, si la fonction compose un bouquet constitué de 5 tulipes, 3 roses et 1 tige de bambou, il doit diminuer le stock de 5 tulipes, 3 roses et 1 tige de bambou et retourner `true`, ou afficher un message et retourner `false` si le stock est insuffisant pour composer le bouquet.

Suite de la réponse :

suite au dos ☞

Question 3 – Trouver les erreurs [26 points]

Le programme fourni ci-dessous contient huit erreurs, dont six sont détectées par le compilateur (page suivante), une lors de l'édition de liens (page suivante)et une se voit lors de l'exécution du programme (page 11).

On vous demande de corriger ces erreurs en les marquant directement sur le code ci-dessous, ou juste en dessous.

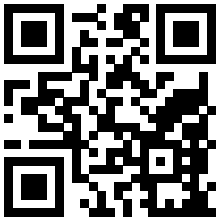
Attention ! On ôtera 2 points pour toute indication d'une erreur qui n'en est pas une.

Afin de facilité votre réponse/lecture, nous fournissons à nouveau le même code en page 13). Vous pouvez aussi répondre là-bas, ainsi que donner des explications complémentaires en bas de page 11.

```
1   #include <iostream>
2   #include <cmath>
3   using namespace std;
4
5   class A {
6   public:
7     void f(double a) const
8     { cout << a + 7.1 << endl; }
9   };
10
11  class B {
12  public:
13    B(double x = 3.14) : b(x) {}
14    B(const B& b2) : b(b2.get_b()) {
15      cout << "Copie d'un B(" << b
16           << ")." << endl;
17    }
18    double get_b() { return b; }
19    void set_b(double x) { b = x; }
20    void f() const {
21      cout << 12.1 + 2.2 * b * b << endl;
22    }
23    virtual void print() const;
24  protected:
25    double b;
26  }
27
28  class C : public A, public B {
29  public:
30    C() = default;
31    C(const C& c2) {
32      cout << "Copie d'un C : "
33           << c2.get_b() << " -> "
34           << b << endl;
35    }
```

```
36    void increment() const { ++b; }
37    void print() const {
38      cout << "je suis un C(" << get_b()
39           << ")" << endl;
40    }
41    void print_equal(B const& b2) const {
42      if (b = b2.b) {
43        cout << "OK, pareils." << endl;
44      } else {
45        cout << "Différents !" << endl;
46      }
47    }
48  };
49
50  int main() {
51    C un_c;
52    un_c.set_b( 2.71 );
53
54    C un_autre_c(un_c);
55    un_autre_c.print();
56    un_autre_c.print_equal(un_c);
57
58    un_autre_c.increment();
59    un_autre_c.print();
60
61    un_autre_c.f( 44.44 );
62
63    return 0;
64  }
```

Ne pas écrire dans cette zone.



Messages du compilateur :

```
erreur2.cc:26:2: error: expected ';' after class definition
   26 | }
      |  ^
      |  ;
erreur2.cc: In copy constructor 'B::B(const B&)':
erreur2.cc:14:30: error: passing 'const B' as 'this' argument discards qualifiers [-fpermissive]
   14 |   B(const B& b2) : b(b2.get_b()) {
      |                     ~~~~~~~~~^~
erreur2.cc:18:10: note:   in call to 'double B::get_b()'
   18 |   double get_b() { return b; }
      |          ^~~~~
erreur2.cc: In copy constructor 'C::C(const C&)':
erreur2.cc:33:21: error: passing 'const C' as 'this' argument discards qualifiers [-fpermissive]
   33 |          << c2.get_b() << " -> "
      |             ~~~~~~~~~^~
erreur2.cc:18:10: note:   in call to 'double B::get_b()'
   18 |   double get_b() { return b; }
      |          ^~~~~
erreur2.cc: In member function 'void C::increment() const':
erreur2.cc:36:30: error: increment of member 'B::b' in read-only object
   36 |   void increment() const { ++b; }
      |                              ^
erreur2.cc: In member function 'virtual void C::print() const':
erreur2.cc:38:37: error: passing 'const C' as 'this' argument discards qualifiers [-fpermissive]
   38 |     cout << "je suis un C(" << get_b()
      |                                 ~~~~~~^~
erreur2.cc:18:10: note:   in call to 'double B::get_b()'
   18 |   double get_b() { return b; }
      |          ^~~~~
erreur2.cc: In member function 'void C::print_equal(const B&) const':
erreur2.cc:42:16: error: 'double B::b' is protected within this context
   42 |     if (b = b2.b) {
      |                ^
erreur2.cc:25:10: note: declared protected here
   25 |   double b;
      |          ^
erreur2.cc:42:11: error: assignment of member 'B::b' in read-only object
   42 |     if (b = b2.b) {
      |         ~~^~~~~~
erreur2.cc: In function 'int main()':
erreur2.cc:61:14: error: request for member 'f' is ambiguous
   61 |   un_autre_c.f( 44.44 );
      |              ^
erreur2.cc:20:8: note: candidates are: 'void B::f() const'
   20 |   void f() const {
      |        ^
erreur2.cc:7:8: note:                 'void A::f(double) const'
    7 |   void f(double a) const
      |        ^
```

suite au dos ☞

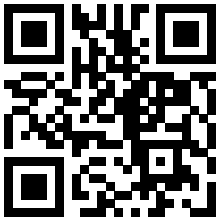
Messages de l'éditeur de liens (une fois les six erreurs de compilation corrigées) :

```
/bin/ld : /tmp/ccn2Qs4t.o : dans la fonction « B::B(double) » :
erreur2_vtable.cc:(.text._ZN1BC2Ed[_ZN1BC5Ed]+0x10) : référence indéfinie vers « vtable for B »
/bin/ld : /tmp/ccn2Qs4t.o : dans la fonction « B::B(B const&) » :
erreur2_vtable.cc:(.text._ZN1BC2ERKS_[_ZN1BC5ERKS_]+0x13) : référence indéfinie vers \
« vtable for B »
/bin/ld : /tmp/ccn2Qs4t.o:(.data.rel.ro._ZTI1C[_ZTI1C]+0x28) : référence indéfinie vers \
« typeinfo for B »
```

Déroulement du programme (une fois les erreurs de compilation et d'édition de liens corrigées) :

```
Copie d'un C : 2.71 -> 3.14
je suis un C(3.14)
Différents !
je suis un C(4.14)
51.54
```

Réponses / explications :



```
1   #include <iostream>
2   #include <cmath>
3   using namespace std;
4
5   class A {
6   public:
7     void f(double a) const
8     { cout << a + 7.1 << endl; }
9   };
10
11  class B {
12  public:
13    B(double x = 3.14) : b(x) {}
14    B(const B& b2) : b(b2.get_b()) {
15      cout << "Copie d'un B(" << b
16           << ")." << endl;
17    }
18    double get_b() { return b; }
19    void set_b(double x) { b = x; }
20    void f() const {
21      cout << 12.1 + 2.2 * b * b << endl;
22    }
23    virtual void print() const;
24  protected:
25    double b;
26  }
27
28  class C : public A, public B {
29  public:
30    C() = default;
31    C(const C& c2) {
32      cout << "Copie d'un C : "
33           << c2.get_b() << " -> "
34           << b << endl;
35    }
36    void increment() const { ++b; }
37    void print() const {
38      cout << "je suis un C(" << get_b()
39           << ")" << endl;
40    }
41    void print_equal(B const& b2) const {
42      if (b = b2.b) {
43        cout << "OK, pareils." << endl;
44      } else {
45        cout << "Différents !" << endl;
46      }
47    }
48  };
49
50  int main() {
51    C un_c;
52    un_c.set_b( 2.71 );
53
54    C un_autre_c(un_c);
55    un_autre_c.print();
56    un_autre_c.print_equal(un_c);
57
58    un_autre_c.increment();
59    un_autre_c.print();
60
61    un_autre_c.f( 44.44 );
62
63    return 0;
64  }
```

Ne pas écrire dans cette zone.

suite au dos ☞

Question 4 – Exécution de programme [30 points]

Le programme suivant compile et s'exécute sans erreurs. Qu'affiche-t-il ?
Répondez-ci dessous puis justifiez (page ci-contre) votre réponse en expliquant les points *importants* (on ne vous demande pas ici de paraphraser le code, mais bien de montrer que vous avez compris ce qui se passe !).

```
#include <iostream>
#include <vector>
using namespace std;

class A {
public:
  A(int x = 2, int y = 3) : a(x), b(y) {}
  int f1() const { return 4*a; }
virtual int f2() const { return 5*b; }
protected:
  int a;
  int b;
};

class B : virtual public A {
public:
  B(int x = 6, int y = 7, int z = 8)
   : A(x,y), c(z) {}
  int f1() const { return  9*a; }
  int f2() const { return  b + 2*c; }
private:
  int c;
};

class C : virtual public A {
public:
  C(int x = 1, int y = 5) : A(x,y) {}
  int f1() const { return 7*a; }
virtual int f2() const { return 11*b; }
};

class D : public B, public C {
public:
  int f1() const { return B::f1(); }
  int f2() const { return C::f2(); }
};
```

```
int main() {
  A a(3, 5);
  A* pa(&a);
  cout << "A) " << a.f1()   << endl;
  cout << "B) " << a.f2()   << endl;
  cout << "C) " << pa->f1() << endl;
  cout << "D) " << pa->f2() << endl;
  B b;
  pa = &b;
  cout << "E) " << b.f1()   << endl;
  cout << "F) " << b.f2()   << endl;
  cout << "G) " << pa->f1() << endl;
  cout << "H) " << pa->f2() << endl;
  C c;
  pa = &c;
  cout << "I) " << c.f1()   << endl;
  cout << "J) " << c.f2()   << endl;
  cout << "K) " << pa->f1() << endl;
  cout << "L) " << pa->f2() << endl;
  D d;
  pa = &d;
  B* pb(&d);
  cout << "M) " << d.f1()   << endl;
  cout << "N) " << d.f2()   << endl;
  cout << "O) " << pa->f1() << endl;
  cout << "P) " << pa->f2() << endl;
  cout << "Q) " << pb->f1() << endl;
  cout << "R) " << pb->f2() << endl;
  a = d;
  pa = &a;
  cout << "S) " << a.f1()   << endl;
  cout << "T) " << a.f2()   << endl;
  cout << "U) " << pa->f1() << endl;
  cout << "V) " << pa->f2() << endl;
  return 0;
}
```

Réponses :

A)
B)
C)
D)

E)
F)
G)
H)

I)
J)
K)
L)

M)
N)
O)
P)
Q)
R)

S)
T)
U)
V)

Ne pas écrire dans cette zone.

Explications :

Place supplémentaire pour répondre à n'importe quelle question si nécessaire. Mais VEUILLEZ INDIQUER LE NUMÉRO DE LA QUESTION TRAITÉE.