EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

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

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

NOM : Hanon Ymous
(000000) #0000
Place : 0

Programmation II (SMA/SPH) : Examen final

25 mai 2018

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 (15h15 - 17h00).
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**.
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, 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 110) :
   1. questions de cours : 22 points ;
   2. conception : 33 points ;
   3. déroulement de programme : 30 points et
   4. chercher les erreurs : 25 points.

   Tous les exercices comptent pour la note finale.

Question 1 – Questions de cours [22 points]

① **[1.5 points]** Le code suivant *compile*-t-il (oui/non) ?

```
class Personne {
public:
    Personne(const string& un_nom) : nom(un_nom) {}
    void dit_bonjour(Personne quidam) {
        quidam.nom = "Pierre";
        cout << nom << " dit bonjour à " << quidam.nom << endl;
    }
private:
    string nom;
};
```

Réponse (0.5 point) :

Pourquoi ? ***Brève* justification (1 point) :**

② **[1.5 points]** Le code suivant *compile*-t-il (oui/non) ?

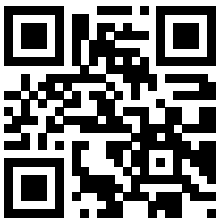
```
class Personne {
public:
    Personne(const string& un_nom) : nom(un_nom) {}
protected:
    string nom;
};

class Eleve : public Personne
{
public:
    Eleve(const string& un_nom) : Personne(un_nom) {}
};

class Prof : public Personne
{
public:
    Prof(const string& un_nom) : Personne(un_nom) {}
    void dit_bonjour(Eleve quidam) {
        cout << nom << " dit bonjour à " << quidam.nom << endl;
    }
};
```

Réponse (0.5 point) :

Pourquoi ? ***Brève* justification (1 point) :**



③ **[1.5 points]** Le code suivant *compile*-t-il (oui/non) ?

```
class Personne {
public:
    Personne(const string& un_nom) : nom(un_nom) {}
protected:
    string nom;
};

class Eleve : public Personne
{
public:
    Eleve(const string& un_nom) : Personne(un_nom), note(0.0) {}
protected:
    double note;
};

class Prof : public Personne
{
public:
    Prof(const string& un_nom) : Personne(un_nom) {}
    void note(Eleve& quidam, double note) {
        quidam.note = note;
        cout << nom << " met la note " << note << endl;
    }
};
```

Réponse (0.5 point) :

Pourquoi ? ***Brève* justification (1 point) :**

suite au dos ☞

④ **[4 points]** Toutes les *portions* de code suivantes sont correctes en C++. Expliquez pour chacune, considérée indépendemment, où l'on peut l'écrire (donnez un contexte valide) et à quoi elle correspond. Donnez à chaque fois un exemple *illustratif*/*pertinent* de même nature (en changeant les noms « `Bidule` » et « `b` » par quelque chose de plus approprié et en donnant un peu plus de contexte pertinent si nécessaire).

	Explication	Exemple *illustratif*
`Bidule b;`	déclaration d'une variable de nom `b` et de type `Bidule`.	`int i;`
`Bidule b();`		
`Bidule b(3);`		
`Bidule b() = 0;`		
`Bidule b = Bidule()`		

⑤ **[3 points]** Avec les déclarations suivantes :

```
class C {
public:
  C();
  C(C const&);
  C(int);
};

void m1(C const& x);
void m2(C x);

C y;
```

a) Quel constructeur est utilisé lors de l'appel `m1(y)` ?

1) aucun **2)** `C::C()` **3)** `C::C(C const&)` **4)** `C::C(int)`

Justifiez *brièvement* votre réponse :

b) Quel constructeur est utilisé lors de l'appel `m2(y)` ?

1) aucun **2)** `C::C()` **3)** `C::C(C const&)` **4)** `C::C(int)`

Justifiez *brièvement* votre réponse :

suite au dos ☞

⑥ **[4 points]** Qu'affiche le code suivant ?

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

class A {
public:
   A(int a) : a(a)
   { cout << "A construit" << endl; }
   ~A()
   { cout << "A detruit" << endl; }
private:
   int a;
};

class B : public A {
public:
   B(int a) : A(a)
   { cout << "B construit" << endl; }
   ~B()
   { cout << "B detruit" << endl; }
};

int main()
{
   A* a(new B(2));
   delete a;
   return 0;
}
```

Réponse (2 points) :

***Brève* justification (2 points) :**

⑦ **[6.5 points]** Avec les déclarations suivantes :

```
class X {
public:
    virtual int a() { return 4   ; }
            int b() { return 3   ; }
    virtual int c() { return b() ; }
};

class Y : public X {
public:
    virtual int a() { return 2   ; }
            int b() { return 1   ; }
    virtual int c() { return a() ; }
};

X x;
Y y;
X* u(&x);
X* v(&y);
Y* w(&y);
```

quelle est la valeur de chacune des expressions suivantes (la réponse peut être 1, 2, 3, 4 ou erreur) ? Justifiez *brièvement* chaque réponse.

`u->b()` :

`v->a()` :

`u->c()` :

`w->c()` :

`v->b()` :

`v->c()` :

Question 2 – Conception OO et programmation [33 points]

Note : in order to help students not so fluent in French, we provide (in parenthesis and in slanted font) an English translation of the least common words. **[end of note.]**

Cette question porte sur la conception d'une partie d'un programme de gestion de dons (*donation management system*) pour une association (*charity organization*). L'architecture que l'on vous demande de concevoir pour ce programme doit minimiser la duplication de code et les dépendances entre parties de code.

Une association a besoin d'un système informatique pour l'aider dans sa gestion de dons (*donation managment*). Cette association gère des lots (*bundle*). Un lot (*bundle*) est simplement un ensemble de dons (*donation set*), caractérisé (le lot) par un donateur (*sponsor*) et une date.

Un don peut être fait en nature (*donation in kind*) ou en espèces (argent ; *money donation*). Un don en nature (*donation in kind*) est caractérisé par :
- le nom du produit (par exemple : « riz (*rice*) ») ;
- sa date de péremption (*expiration date*) ;
- sa quantité (en kg) ;
- et le prix au kg.

Un don en espèces (*money donation*) est caractérisé par :
- son montant (*amount*) ;
- sa devise (*currency*).

Voici les règles utilisées pour l'évaluation d'un don (*to evaluate/classify a donation*) :

1. La valeur d'un don :
   - en nature (*donation in kind*) est donnée par la quantité multipliée par le prix du kg ;
   - en espèces (*money donation*) correspond simplement à son montant.
2. Tout don peut être invalide (*null/unwanted*) :
   - un don en nature n'est pas valide si sa date de péremption est dépassée ;
   - un don en espèces n'est pas valide si sa devise n'est pas autorisée.

   On suppose que tout objet du code peut avoir accès à la date du jour ou à la liste des devises autorisées au travers de fonctions globales.

L'association souhaite de plus avoir le moyen de :

1. calculer la *valeur* totale des dons valides (*wanted/not null*) d'un lot (*bundle*) ;
2. connaître le *nombre* de dons invalides (*null/unwanted*) d'un lot.

① **(28 points)** Sans donner le détail du code lui-même (on ne demande ici qu'une *conception*), écrivez **en C++** les classes, les 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 de méthodes).

Indiquez également les constructeurs et les destructeurs lorsque nécessaire.

② **(5 points)** Donnez le code complet (définition) de la méthode ou fonction permettant de calculer la valeur totale des dons valides d'un lot donné.

Réponse :

Suite de la réponse à la Question 2 :

Suite de la réponse à la Question 2 :

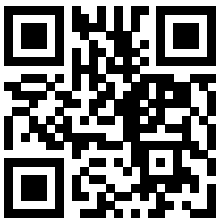
Question 3 – Déroulement de programme [30 points]

Le programme ci-contre compile et s'exécute sans erreurs. Qu'affiche-t-il ?
Répondez ci-dessous puis justifiez votre réponse en expliquant les points *importants* (on ne vous demande pas ici de paraphraser le code, ni de justifier chaque affichage individuellement, mais bien de montrer que vous avez compris ce qui se passe ! Il n'est pas nécessaire de remplir plus que la place laissée ci-dessous. Vous pouvez, si nécessaire, annoter le code ci-contre).

Réponse (19 points) :

A)	H)	N)
B)	I)	O)
C)	J)	P)
D)	K)	Q)
E)	L)	R)
F)	M)	S)
G)		

Justifications (11 points) :



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

class X {
  public:
    X() : x(4.2)    { cout << "X() "   ; }
    virtual ~X()    { cout << "~X() "  ; }
    X(double y) : x(y) { cout << "X(double) "; }
    void set(double y) { x = y; }
    double get() const { return x; }
    void print() const { cout << get() << " " ; }
  private:
    double x;
};

class Z : virtual public X {
  public:
    Z()          { cout << "Z() " ; }
    virtual ~Z() { cout << "~Z() "; }
    void f() const { cout << "Z::f "; }
    virtual void g() const { cout << "Z::g "; }
};

class Y : virtual public X {
  public:
    Y()          { cout << "Y() " ; }
    virtual ~Y() { cout << "~Y() "; }
};

class W : public Y, public Z {
  public:
    W()          { cout << "W() " ; }
    virtual ~W() { cout << "~W() "; }
    void f() const { cout << "W::f "; }
    void g() const { cout << "W::g "; }
};

void f1(X  x) { x.set( x.get()  + 1.1); }
void g1(X& x) { x.set( x.get()  + 2.2); }
void h1(X* x) { x->set(x->get() + 3.3); }

X f2(X  x) { x.set(x.get() + 4.4); return x; }
X g2(X& x) { x.set(x.get() + 5.5); return x; }
X h2(X* x) {
  X x1 = *x;
  x->set( x->get() + 6.6 );
  return x1;
}
```

```
int main() {
  cout << "A) ";
  Y c;
  cout << endl << "B) ";
  Z* b = new W;
  cout << endl << "C) ";
  b->g();
  cout << endl << "D) ";
  b->f();

  cout << endl << "E) ";
  X a = X(0);
  cout << endl << "F) ";
  f1(a);
  cout << endl << "G) ";
  a.print();
  cout << endl << "H) ";
  g1(a);
  cout << endl << "I) ";
  a.print();
  cout << endl << "J) ";
  h1(&a);
  cout << endl << "K) ";
  a.print();
  cout << endl << "L) ";
  a = f2(a);
  cout << endl << "M) ";
  a.print();
  cout << endl << "N) ";
  a = g2(a);
  cout << endl << "O) ";
  a.print();
  cout << endl << "P) ";
  a = h2(&a);
  cout << endl << "Q) ";
  a.print();

  cout << endl << "R) ";
  delete b;
  cout << endl << "S) ";
}
```

Question 4 – Chercher les erreurs [25 points]

Le programme fourni ci-contre contient de nombreuses erreurs dont certaines sont détectées par le compilateur (messages ci-dessous), d'autres non, et dont une pourrait causer un arrêt prématuré de l'exécution du programme.

Indiquez directement sur le code ci-contre toutes les erreurs, en donnant à chaque fois dans la marge droite une *explication* de l'erreur. Si cela fait sens, proposer également une correction.

Attention ! On ôtera 1 point pour toute indication d'une erreur qui n'en est pas une.

Voici les messages d'erreurs du compilateur :

```
debug.cc:18:9: error: conflicting return type specified for
    'virtual int B::f2(std::ostream&) const'
debug.cc:8:18: error:   overriding 'virtual void A::f2(std::ostream&) const'
debug.cc: In function 'int main()':
debug.cc:38:5: error: cannot declare variable 'a' to be of abstract type 'A'
debug.cc:4:7: note:   because the following virtual functions are pure within 'A':
debug.cc:7:18: note:  virtual void A::f1() const
debug.cc:39:5: error: use of deleted function 'B::B()'
debug.cc:14:7: note: 'B::B()' is implicitly deleted because the default definition
    would be ill-formed:
debug.cc:14:7: error: no matching function for call to 'A::A()'
debug.cc:6:5: note: candidate: A::A(int)
debug.cc:6:5: note:   candidate expects 1 argument, 0 provided
debug.cc:4:7: note: candidate: constexpr A::A(const A&)
debug.cc:4:7: note:   candidate expects 1 argument, 0 provided
debug.cc:4:7: note: candidate: constexpr A::A(A&&)
debug.cc:4:7: note:   candidate expects 1 argument, 0 provided
debug.cc:41:6: error: could not convert '* c' from 'A' to 'B'
debug.cc:42:13: error: invalid operands of types 'void' and
    '<unresolved overloaded function type>' to binary 'operator<<'
debug.cc:46:6: error: request for member 'f3' in 'b2', which is of non-class type 'B()'
debug.cc:47:5: error: request for member 'f2' in 'c', which is of pointer type 'A*'
    (maybe you meant to use '->' ?)
debug.cc:48:8: error: 'void B::f3()' is private within this context
debug.cc:15:10: note: declared private here
```

```
1  #include <iostream>
2  using namespace std;
3
4  class A {
5    public:
6      A(int x) : x(x) {}
7      virtual void f1() const = 0;
8      virtual void f2(ostream& out) const
9        { out << "A::f2" << endl; }
10   private:
11     int x;
12 };
13
14 class B : public A {
15     void f3() { cout << "B::f3" << endl; }
16   public:
17     void f1() const { cout << "B::f1" <<endl; }
18     int f2(ostream& out) const
19       { out << "B::f2" << endl; return 0; }
20 };
21
22 void operator<<(ostream& out, const B & b)
23 { b.f2(out); }
24
25 class C : public A {
26   public:
27     C() : A(0), y(0) {}
28     void f2(ostream& out)
29       { out << "C::f2" << endl; }
30   private:
31     void f1() const { cout <<"C::f1" <<endl; }
32     int y;
33 };
34
35 void f2(B b) { b.f2(cout); }
36
37 int main() {
38   A a(1);
39   B b;
40   A* c = new C();
41   f2(*c);
42   cout << b << endl;
43   A* a1;
44   a1->f2(cout);
45   B b2();
46   b2.f3();
47   c.f2(cout);
48   b.f3();
49 }
```