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

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

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

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

Programmation I (SMA/SPH) : EXAMEN Semestre 1

21 décembre 2017

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 (10h15 – 12h00).
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 cinq 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 107) :
   1. questions de cours : 17 points ;
   2. conception : 33 points ;
   3. programmation : 10 points ;
   4. déroulement de programme : 17 points et
   5. chercher les erreurs : 30 points.

   Tous les exercices comptent pour la note finale.

Question 1 – Questions de cours [17 points]

Question 1.1 [10 points]

Est-il licite d'écrire les portions de code suivantes dans un programme (indépendamment les unes des autres), éventuellement dans une expression plus large ? Si oui, dire à quoi cela correspond.

	correct ? (oui/non)	correspond à :
`a = bool;`		
`b = true;`		
`true = c;`		
`(false == d)`		
`(e == true)`		
`vector<double> u();`		
`double f(10);`		
`vector<double> v(vector<int>);`		
`vector<int> w(bool) = false;`		
`vector<double> x(int = 2);`		
`vector<char> y(10);`		
`typedef vector<int> Ligne(10);`		

Question 1.2 [3 points]

Quels sont les intérêts de créer ses propres types ? Donnez un exemple.

Question 1.3 [4 points]

En C++, par quelles étapes faut-il passer pour écrire du texte dans un fichier ?
Illustrez par un exemple simple.

suite au dos ☞

Question 2 – Conception [33 points]

Un service de l'EPFL possède la liste de tous les étudiants stockée dans un seul et même fichier. À chaque étudiant est attribuée une série d'informations :
numéro d'identification ; nom ; prénom ; date de naissance ; adresse ; section (p. ex. IN, MA, PH) ; et cycle d'étude (p. ex. BA, MA, PHD).

Pour chaque étudiant, les informations sont stockées ligne par ligne dans le format (un peu flexible) suivant :

Format :

```
ID
<numéro d'identification>
FIRSTNAME
<prénom>
LASTNAME
<nom>
ADDRESS
<adresse>
BIRTHDATE
<année de naissance>
SECTION
<section>
LEVEL
<cycle d'étude>
###
```

où « `###` » est un séparateur (de fin) entre les différents étudiants.
Notez bien que l'ordre des couples de lignes « (LABEL, valeur) » est quelconque (p.ex. (`ID`, numéro d'identification), (`FIRSTNAME`, prénom), etc., n'arrivent pas toujours au même endroit).
Voir l'exemple ci-contre.

Exemple de fichier :

```
LASTNAME
Clinton
ID
123245
FIRSTNAME
Bill
ADDRESS
WhiteHouse, 1200 Washington DC
BIRTHDATE
1980 04 01
SECTION
IN
LEVEL
PHD
###
FIRSTNAME
John
LEVEL
MA
LASTNAME
Doe
ID
456456
ADDRESS
Av. du Mail 12, 1205 Geneve
BIRTHDATE
1982 07 25
SECTION
MA
###
```

Il vous est demandé de concevoir un programme permettant de gérer de telles listes d'étudiants :

① **[7 points]** Définir les types de données permettant de gérer de telles informations.
Expliquer vos choix d'implémentation.

② **[9 points]** Écrire le prototype et le corps de la fonction `load` permettant de charger en mémoire la liste des étudiants, à partir du nom du fichier.

Il est très fortement conseillé d'utiliser une approche *modulaire*, en particulier pour tout ce qui concerne les accès fichiers. On ***ne*** vous demande ***pas*** d'écrire le code de vos éventuelles sous-fonctions, mais simplement d'expliquer en une ligne de commentaire ce qu'elles font.

③ **[9 points]** Écrire le prototype et le corps de la fonction `pourcentage` qui retourne le pourcentage d'étudiants d'une section et d'un cycle d'étude passés en paramètres parmi la population totale des étudiants (déjà chargée en mémoire).

On vous demande ici le code détaillé complet (*pas* de modularisation ici).

④ **[8 points]** Écrire le prototype et le corps de la fonction `saveAddress`. Cette fonction reçoit une liste d'étudiants et sauvegarde dans un fichier, le nom, le prénom et l'adresse de chaque étudiant d'une section passée en paramètre. Le fichier où sauvegarder ces informations sera nommé selon la convention suivante : *`SECTION`*`.dat`. Par exemple, l'étudiant John Doe de l'exemple ci-contre sera sauvegardé dans le fichier « `MA.dat` ».

Les informations de chaque étudiant seront sauvegardées sur une seule ligne et séparées par des points-virgules (`;`).

On vous demande ici le code détaillé complet (*pas* de modularisation ici).

(suite de la réponse si nécessaire)

(suite de la réponse si nécessaire)

suite au dos ☞

Question 3 – Programmation [10 points]

① **[5 points]** Expliquer ce que fait la fonction `bidule` suivante :

```
int bidule(int u, int v) {
  if (v <= 1) return u;
  else return bidule(u*v, v-1);
}
```

② **[5 points]** Donner la définition d'une fonction `machin` ayant le même comportement que `bidule`, mais *sans* appel récursif.

Question 4 – Déroulement de programmes [17 points]

Question 4.1 [4 points]

Le programme suivant compile sans erreur. Qu'affiche-t-il ?
Expliquez *brièvement* les principaux aspects.

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

double a(double& x) {
  x = x + 1.3;
  return x;
}

int main() {
  double u(4.1);

  double v(a(u));

  double w(a(v));

  cout << u << ", "
       << v << ", "
       << w << endl;

  return 0;
}
```

suite au dos ☞

Question 4.2 [5 points]

Le programme suivant compile sans erreur. Qu'affiche-t-il ?
Expliquez *brièvement* les principaux aspects.
(Vous pouvez commencer à répondre à droite du code)

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

int a(25);

int f(int b) {
  return a * b;
}

int g(int a, int b) {
  b = (b + a) / 4 ;
  return b;
}

int main() {
  int a(5);
  int b(3);
  a = g( f(a), f(b) );
  cout << a << "; " << b << endl;
  return 0;
}
```

Question 4.3 [8 points]

Le programme suivant compile sans erreur. Déterminez son résultat *en détaillant les étapes* de calcul. Il est conseillé de s'aider du tableau ci-contre.

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

double f(double a) { return 3.0 * a; }

int main()
{
  int a(9);
  int b(-1);

  while (a > -3) {
    if (b > 0) {
      a -= 4 * f(b);
      b--;
    } else {
      b++;
    }

    cout << "a=" << a << ", b=" << b << endl;
  }

  cout << "En fin : ";
  if (a < b) cout << a; else cout << b;
  cout << endl;
}
```

a	b	affiche

Explications complémentaires :

suite au dos ☞

Question 5 – Chercher les erreurs [30 points]

Question 5.1 [10 points]

On considère le programme C++ suivant, qui compile :

```
1   #include <iostream>
2   using namespace std;
3
4   bool f(int a);
5
6   bool g(int a) {
7     if (a == 0) return true;
8     else return f(a-1);
9   }
10
11  bool f(int a) {
12    if (a == 1) return true;
13    else return g(a-1);
14  }
15
16  int main() {
17    cout << "Entrez un nombre entier : ";
18    int a;
19    cin >> a;
20
21    cout << a << " est ";
22    if (f(a)) cout << "...(A)...";
23    if (g(a)) cout << "...(B)...";
24    cout << endl;
25
26    return 0;
27  }
```

① **[1 point]** Quel concept important met-il en œuvre ?

② **[3 points]** Indiquer le but et le fonctionnement général attendu de ce programme (c.-à-d. l'intention du programmeur). Que mettre à la place de « `(A)` » et de « `(B)` » ?

③ **[6 points]** Ce programme fonctionne-t-il correctement (mis à part d'éventuels problèmes de saisie, non considérés ici) ? Si oui, le justifier/démontrer. Si non, donner les raisons principales et corriger le programme en essayant de changer le moins de code possible et en gardant l'esprit d'origine.
(Vous pouvez répondre sur le code lui-même, à droite de celui-ci ou sur la page ci-contre.)

(suite de la réponse si nécessaire)

suite au dos ☞

Question 5.2 [12 points]

Le programme suivant donne 5 essais pour trouver la valeur de 1 euro en francs suisses. Malheureusement, il comporte des erreurs. Trouvez-les, expliquez-les et corrigez-les (au choix : soit directement sur le code, soit en dessous et sur la page ci-contre).
On ôtera 1 point pour toute indication d'une erreur qui n'en est pas une.

```
1   #include <iostream>
2   #include <cmath>
3   using namespace std;
4
5   constexpr double taux_change;
6   constexpr int nb_essais;
7   taux_change = 1.18;
8   nb_essais = 5;
9
10  bool egaux(double x, double y, double precision = 1e-2)
11  {
12    return abs(x - y) < precision;
13  }
14
15  int main()
16  {
17    int x;
18    cout << "Quelle est la valeur de 1 euro en francs suisses ?" << endl;
19    bool trouve;
20    for (int i(1); trouve or (i < nb_essais); ++i) ;
21    {
22         cout << "Essai no" << i << " : ";
23         cin >> x;
24         trouve = egaux(x, taux_change);
25    }
26    if (i > 10)
27      cout << "Perdu.  :-(" << endl;
28    else
29      cout << "Bravo ! :-)" << endl;
30
31    return 0;
32  }
```

(suite de la réponse si nécessaire)

suite au dos ☞

Question 5.3 [8 points]

Le programme suivant est censé afficher le produit des nombres entiers compris entre 11 et 24 (inclus). Cependant il comporte quelques erreurs. Trouvez-les, expliquez-les et corrigez-les.
On ôtera 1 point pour toute indication d'une erreur qui n'en est pas une.

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       double x;
6       double resultat;
7
8       for (x >= 11.0 ; x <= 24.0 , ++x)
9           resultat = resultat * x;
10
11      cout << "Produit : resultat" << endl;
12
13      return 0;
14  }
```