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

Faculté Informatique et Communication
Cours d'Informatique à la section SV
Sam J.

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

PROGRAMMATION I

Examen Semestre I

Instructions :

— Vous disposez de une heure quarante cinq minutes pour faire cet examen (15h15 - 17h).

— Nombre maximum de points: 115 points (dont 20 en bonus).

— Indiquez votre **NUMÉRO SCIPER** sur *chacune* des feuilles. **Une feuille sans identification ne sera pas corrigée.**

— Toute documentation est autorisée, hormis les corrigés des anciens tests ;

— Répondez sur les feuilles qui vous sont distribuées à cet effet (utilisez aussi le verso des feuilles si nécessaire). **Ne répondez pas sur l'énoncé**.

— Vous pouvez répondre aux questions en français ou en anglais.

— Veillez à **ne traiter *qu'un* exercice par feuille**.

— L'examen compte 3 exercices indépendants.

Vous pouvez commencer par celui que vous souhaitez

— Exercice 1: **65** points.
— Exercice 2: **30** points.
— Exercice 3: **20** points.

SUJET À LA PAGE SUIVANTE

SUJET À LA PAGE SUIVANTE

Exercice 1 : Conception de programme et programmation [65 points]

Un éleveur de rennes vous sollicite de sa lointaine Laponie pour que vous l'aidiez à gérer son cheptel.

Il s'agit notamment de vacciner les animaux pour les prémunir des maladies de l'hiver.

Rennes et vaccins

Un *renne* est caractérisé par :

1. son *matricule* (un ensemble de chiffres et de lettres);
2. son âge;
3. son poids;
4. son sexe;
5. l'ensemble des vaccins qu'il a reçu.

Un *vaccin* est caractérisé par son nom (une chaîne de caractères). Un cheptel est un ensemble de rennes.

Question 1 : structures de données [8 points]

Donner un code C++ possible pour les structures de données permettant de modéliser :

1. Un renne.
2. Un vaccin.
3. Un cheptel.

Vous pourrez bien sûr définir d'autres types si nécessaire et ferez un usage judicieux du `typedef`. On rappelle qu'une structure de données n'est pas forcément une `struct` (ce peut être un simple type).

Question 2 : fonctionnalités [42 points]

Les fonctionnalités suivantes sont nécessaires à notre éleveur :

1. Afficher un cheptel, c'est à dire les caractéristiques de chacun de ses rennes.
2. Ajouter un renne donné à un cheptel.
3. Supprimer d'un cheptel un renne de matricule donné.
4. Compter tous les rennes de sexe donné d'un cheptel.
5. Sélectionner (sans l'afficher) le renne d'un cheptel capable de tirer un traîneau: il s'agira d'un renne mâle, dont le poids est supérieur à un seuil minimal donné et dont l'âge n'excède pas une limite donnée.
6. Administrer un vaccin donné à un renne donné.
7. Pour un cheptel donné, administrer un vaccin donné à tous les rennes dans une certaine tranche d'âge.
8. Trouver (sans les afficher) tous les rennes d'un cheptel n'ayant reçu aucun vaccin.
9. Trouver (sans les afficher) tous les rennes d'un cheptel n'ayant pas reçu un vaccin donné.

10. Trouver (sans les afficher) tous les rennes d'un cheptel qui n'ont reçu aucun des vaccins d'un ensemble de vaccins donné.

Donnez le prototype de ces fonctions. On ne vous demande pas d'écrire un programme complet ou le corps des fonctions mais uniquement les prototypes. Il n'est pas demandé non plus d'écrire des directives d'inclusion (`#include<..>`).

Question 3 : programmation [15 points]

Donnez le code des fonctions mettant en oeuvre les fonctionnalités 5 et 10.
Vous utiliserez les fonctionnalités prototypées sans avoir besoin d'en donner les corps. Veillez à respecter une bonne indentation. Vous pouvez ajouter de nouvelles fonctions si cela permet de mieux modulariser. **Dans ce cas vous donnerez aussi leur corps**.

suite au verso ☞

Exercice 2 : Questions de cours [30 points]

Répondez clairement et succinctement aux questions suivantes :

① [**4 points**] Soit le programme suivant (on suppose que toutes les inclusions nécessaires sont faites) :

```
0.  int main()
1.  {
2.      cout << "Quel temps fait-il ?" << endl;
3.      string answer;
4.      cin >> answer;

5.      while (not (answer == "soleil" or
6.                  answer == "pluie")) {
7.          cout << "Quel temps fait-il ?" << endl;
8.          cin >> answer;
9.      }

10.     return 0;
11. }
```

(a) Que se passe t'il avec ce code si on ajoute la ligne `string answer;` juste avant la ligne 8 ? (la ligne apparaitrait alors deux fois dans le code)

(b) Comment proposez-vous de le réécrire pour éviter les répétitions des lignes 2 et 7 et des lignes 4 et 8?(donnez le code C++)

② [**5 points**] Soit le programme suivant (on suppose que toutes les inclusions nécessaires sont faites) :

```
0.  int f(int& i)
1.  {
2.      i = 1;
3.      ++i;
4.      return i;
5.  }

6.  int g(int i)
7.  {
8.      i = 4;
9.      ++i;
10.     return i;
11. }

12. int main()
13. {
14.     int i(5);
15.     cout << f(i) + g(i) + i << endl;

16.     return 0;
17. }
```

Qu'affiche t-il ? expliquez brièvement son déroulement.

③ [**6 points**] Soit le programme suivant (on suppose que toutes les inclusions nécessaires sont faites). Il est censé trouver l'indice de la première valeur paire dans un tableau :

```
0.  typedef vector<int> Tableau;

1.  int trouver_pair(Tableau t)
2.  {
3.      int i(-1);
4.      bool found (false);
5.      for (auto val : t) {
6.          if ((val % 2) != 0 or not found)
7.              ++i;
8.          if (val % 2 == 0) found = true;
9.      }
10.     return i;
11. }

12. int main ()
13. {
14.     Tableau t( { 1, 3, 4, 6 , 8, 10 } );

15.     int indice(trouver_pair(t));
16.     if (indice < 0) {
17.         cout << "Il n'y a pas de valeurs paires dans le tableau" << endl;
18.     } else {
19.         cout << "Indice de la première valeur paire : " << indice << endl;
20.     }
21.     return 0;
22. }
```

(a) Pourquoi l'utilisation d'une itération sur ensemble de valeurs n'est-elle pas le choix le plus naturel dans la fonction `trouver_pair`?

(b) Quelle(s) critique(s) pouvez-vous formuler à propos de la fonction `trouver_pair` ?

(c) Comment proposeriez-vous de réécrire cette fonction pour y remédier ?

suite au verso ☞

④ **[4 points]** Soit le code suivant (on suppose que toutes les inclusions nécessaires sont faites) :

```
0.  struct S {
1.      string s;
2.      size_t c;
3.  };

4.  void f(S& s)
5.  {
6.      if (s.c  >= s.s.size()) {
7.          s.c = 0;
8.      }
9.      size_t i(s.c);
10.     s.s[i] = s.s[s.s.size()-i-1];
11.     ++s.c;
12. }

13. int main()
14. {
15.     S s({"royal", 0});
16.     for (size_t i(0); i < s.s.size(); ++i) {
17.         f(s);
18.     }
19.     cout << s.s << endl;
20.     cout << s.c << endl;
21.     return 0;
22. }
```

Qu'affiche t-il ? expliquez brièvement son déroulement.

⑤ **[5 points]** Soit le programme suivant (on suppose que toutes les inclusions nécessaires ont été faites) :

```
0.  void int_to_file(ofstream& o, size_t nb)
1.  {
2.      for (size_t i(0); i < nb; ++i)
3.          o << i << endl;
4.  }

5.  int main()
6.  {
7.      ofstream sortie;

8.      int_to_file(sortie, 10);

9.      return 0;
10. }
```

(a) Comment proposez-vous de le modifier pour qu'il permette de stocker correctement les entiers de 0 à 10 dans un fichier donné?

(b) Est-il possible de remplacer le prototype de `int_to_file`, par le suivant ? (Justifiez votre réponse)

```
void int_to_file(ofstream o, size_t nb)
```

⑥ [**6 points**] Soit le code suivant (on suppose que toutes les inclusions nécessaires sont faites) :

```
0.  const string s("magic ");

1.  void f(string s1, string s2)
2.  {
3.      cout << "f(string, string)" << endl;
4.      throw s + s2 + s1 ;
5.  }

6.  void g(string s)
7.  {
8.      cout << "g(string)" << endl;
9.      try {
10.         f(s, "wonder");
11.     } catch(const string& e) {
12.         throw  e + s;
13.     }
14. }

15. int main()
16. {
17.     try {
18.         g("land ");
19.         f("abbacus ","lost ");
20.     } catch(const string& e) {
21.         cout << e << endl;
22.     }
23.     return 0;
24. }
```

(a) Qu'affiche t-il ? expliquez brièvement son déroulement.

(b) Qu'afficherait-il si l'on inverse les lignes 18 et 19 ? expliquez brièvement.

suite au verso ☞

Exercice 3 : Déroulement de programme [20 points]

Le programme suivant compile et s'exécute sans erreurs (C++11).

```
0.  #include <utility>
1.  #include <vector>
2.  #include <string>
3.  #include <iostream>
4.  using namespace std;

5.  struct Color {
6.     int w;
7.     string* n;
8.  };

9.  typedef vector<Color> T;

10. void g(T& t)
11. {
12.    for (size_t i(0); i < t.size(); ++i)
13.       for (size_t j(0); j < t.size()-i-1;
               ++j)
14.       {
15.         if (t[j].w > t[j+1].w)
16.            swap(t[j], t[j+1]);
17.       }
18. }

19. void d(const T& t)
20. {
21.    if (t.empty()) {
22.       cout << "Colorless world" << endl;
23.       return;
24.    }

25.    for (auto val : t) {
26.       cout << *(val.n) << " ";
27.    }
28.    cout << endl;
29. }
```

```
30. void f(T& t)
31. {
32.    if (t.empty()) return;
33.    int m(t[0].w);
34.    string* s(t[0].n);
35.    size_t k(0);
36.    for (size_t i(1); i < t.size(); ++i) {
37.       if (t[i].w >= m) {
38.         m=t[i].w;
39.         s=t[i].n;
40.         k = i;
41.       }
42.    }
43.    t[k].w = -1;
44.    cout << *s << endl;
45. }

46. int main()
47. {
48.    T t;
49.    t.push_back( { 2, new string("Orange") } );
50.    t.push_back( { 0, new string("Black") } );
51.    t.push_back( { 1, new string("Blue") } );
52.    t.push_back( { 4, new string("Yellow") } );
53.    d(t);
54.    cout << "*****" << endl;
55.    g(t);
56.    d(t);
57.    cout << "*****" << endl;
58.    for (int i(0); i < 4; ++i) {
59.       f(t);
60.    }
61.    cout << "*****" << endl;
62.    t.clear();
63.    d(t);

64.   return 0;
65. }
```

Qu'affiche t-il ? Expliquez succinctement sont déroulement. Il ne s'agit pas ici de paraphraser le code, mais bien d'*expliquer* les étapes et le déroulement du programme (Aidez vous du tableau fourni dans la feuille de réponse).
Expliquez pourquoi ce programme ne gère pas correctement la mémoire et donnez les lignes de codes C++ à ajouter eu où les placer pour résoudre ce problème