Nom : N° d'étudiant :

Examen de programmation C++

— Master Modélisation Aléatoire —

Février 2016, durée 2h.

Le sujet est composé de deux exercices indépendants. Tous les appareils électroniques sont interdits et seule une feuille de memento est autorisée.

Les méthodes demandées à une question peuvent être utilisées aux questions suivantes même si la question n'a pas été traitée.

Lors de l'utilisation d'une classe ou d'une méthode de la STL, il n'est pas nécessaire de donner le nom exact de celle-ci. Il suffit de lui donner un nom quelconque mais explicite et d'expliquer ce que cette classe/méthode est censée faire.

Les réponses aux questions doivent directement être écrites dans les cadres du sujet.

► **Exercice 1** Donner les affichages du programme suivant.

```
class B {
public:
  string f() const { return g() + " by f() in B"; }
  virtual string g() const { return "g() in B"; }
};

class D : public B {
public:
  string f() const { return g() + " by f() in D"; }
  virtual string g() const { return "g() in D"; }
};

int main() {
  cout << B().f() << endl;
  cout << B().g() << endl;
  cout << D().f() << endl;
  cout << D().g() << endl;
  cout << ((B) D()).f() << endl;
}
```


► **Exercice 2** On considère le programme suivant incomplet.

```
// Class C
class C {
public:
  C(int v) : value(v) {}        // Constructor
private:
  int value;                    // Value
};

int main() {
  int t[4] = { 1, 2, 3, 4};                 // Array of 4 integers
  print(t, t+4);                            // Print: 1 2 3 4

  vector<string> v { "He", "l", "l", "o"};  // Array of 4 strings
  print(v.begin(), v.end());                // Print: He l l o

  C r[4] = { C(1), C(2), C(3), C(4) };      // Array of 4 C instances
  print(r, r+4);                            // Print: 1 2 3 4
}
```

L'objectif global de l'exercice est de compléter le programme pour que les affichages marqué après les `Print:` dans les commentaires se fassent correctement.

a) Quels sont les types des valeurs passées à la fonction `print` lors des deux premiers appels à celle-ci.
b) Écrire des fonctions `print` en utilisant la *surcharge* pour les deux premiers appels.
c) Écrire une fonction `print` en utilisant un *template* pour les trois appels.
d) Que faut-il ajouter à la classe `C` pour que le dernier appel à `print` fonctionne avec la fonction écrite à la question précédente.

▶ **Exercice 3** On désire modéliser un système de fichiers contenant uniquement des fichiers et des répertoires. La spécification est la suivante.

+ chaque fichier/répertoire a un identifiant unique (entier). C'est 0 pour le premier fichier/répertoire créé, 1 pour le deuxième et ainsi de suite.
+ un fichier a une longueur (entier)
+ un répertoire contient des associations entre des noms (string) et des fichiers/répertoires
+ chaque fichier/répertoire a une méthode `size()` qui retourne sa *taille*, c'est-à-dire
  - sa longueur pour un fichier
  - son nombre d'associations pour un répertoire
+ chaque fichier/répertoire a une méthode `disk()` qui retourne son *occupation disque*, c'est-à-dire
  - sa longueur pour un fichier
  - pour un répertoire, la somme de sa taille multipliée par 32 et des occupations des fichiers/répertoires qu'il référence.

— Écrire des classes `File` et `Directory` pour les fichiers et les répertoires. Le code suivant doit fonctionner avec les classes écrites. On devra en particulier produire les affichages décrits par les commentaires commençant par `Print:`.

```
int main() {
  File f1(100);                 // Création d'un fichier f1 de taille 100
  cout << f1.size() << endl;
  // Print: 100
  cout << f1.disk() << endl;
  // Print: 100
  Directory d1;                 // Création d'un répertoire vide d1
  cout << d1.size() << endl;
  // Print: 0
  cout << d1.disk() << endl;
  // Print: 0
  d1.add("f1", &f1);            // Ajout de f1 à d1 sous le nom f1
  cout << d1 << endl;
  // Print: Dir 1: f1 --> 0 (100)
  File f2(1000);                // Création d'un fichier f2 de taille 1000
  d1.add("f1", &f2);            // Remplacement de f1 par f2 dans d1
  cout << d1 << endl;
  // Print: Dir 1: f1 --> 2 (1000)
  d1.add("f2", &f1);            // Ajout de f1 à d1 sous le nom f2
  cout << d1 << endl;
  // Print: Dir 1: f1 --> 2 (1000), f2 --> 0 (100)
  cout << d1.size() << endl;
  // Print: 2
  cout << d1.disk() << endl;
  // 1164 = (32 * 2 + 1000 + 100)
  Directory d2;                 // Création d'un répertoire vide d2
  d2.add("f3", &f1);            // Ajout de f1 à d2 sous le nom f3
  d2.add("d1", &d1);            // Ajout de d1 à d2 sous le nom d1
  cout << d2 << endl;
  // Print: Dir 3: d1 --> 1 (2), f3 --> 0 (100)
}
```