Trimestre Automne, 2015

Mohamed Lokbani

IFT1169 – Examen Final –

Inscrivez tout de suite : votre nom et le code permanent.

Nom : _____| Prénom(s) : _____|

Signature : _____| Login : _____|

Date : Mercredi 9 décembre 2015

Durée : 3 heures (de 18h30 à 21h30)

Local : Z-330, Pavillon Claire McNicoll

Directives :

- Toute documentation est permise.
- Calculatrice **non** permise.

- Répondre directement sur le questionnaire.
- Les réponses **doivent être brèves, précises, claires et nettement présentées.**

1._____/08 (1.1, 1.2, 1.3, 1.4)

2._____/12 (2.1, 2.2, 2.3)

3._____/20 (3.1)

4._____/20 (4.1)

5._____/20 (5.1)

Total : _____/100

Directives officielles

* Interdiction de toute communication verbale pendant l'examen.

* Interdiction de quitter la salle pendant la première heure.

* L'étudiant qui doit s'absenter après la première heure remettra sa carte d'étudiant au surveillant, l'absence ne devant pas dépasser 5 minutes. Un seul étudiant à la fois peut quitter la salle.

* Toute infraction relative à une fraude, un plagiat ou un copiage est signalée par le surveillant au directeur de département ou au professeur qui suspend l'évaluation.

F.A.S

Exercice 1 (08 points) répondez par « vrai » ou « faux » en y incluant une très courte explication.

1.1 [VRAI | FAUX] Le paramètre « T » dans « template <class T> » est obligatoirement un type primitif défini par le langage « C++ », comme « int », ou « double », etc.

1.2 [VRAI | FAUX] En « C++ », une exception qui n'est pas capturée est ignorée par le programme.

1.3 [VRAI | FAUX] Cette instruction va générer une erreur à la compilation : `typedef vector<int> X;`

1.4 [VRAI | FAUX] À un instant donné, un stream associé à un fichier peut avoir « stream.good() » à « false » même si « stream.bad() » est, elle aussi à « false ».

Exercice 2 (12 points) Vous avez conçu un jeu en ligne. Chaque fois qu'un joueur termine de jouer, vous allez vérifier s'il a fait le meilleur score. Si c'est le cas, vous allez préserver le score et l'identité du joueur. Vous allez avoir grosso modo le code suivant :

```
int max_score; // le score le plus élevé
int max_id; // id du joueur ayant obtenu le score le plus élevé
void maj(int score, int id){
        if (score > max_score) {
                max_score = score;
                max_id = id;
        }
}
void affiche(){
        cout << "max score est : " << max_score << endl;
        cout << "par le joueur : " << max_id << endl;
}
```

Nous allons examiner ce fragment de code dans le cas de deux joueurs qui jouent en même temps. Nous allons supposer que le système permet la gestion de threads. Chaque thread va exécuter ces tâches de manières concurrentes (bref la notion de threads).

On vous demande de nous expliquer si les situations ci-dessous posent des problèmes. Est-ce que les appels aux fonctions vont fonctionner correctement. Dans tous les cas, on vous demande de justifier votre réponse.

2.1 Les deux threads exécutent la fonction « maj » pour mettre à jour le score de deux différents joueurs.

2.2 Le premier thread exécute la fonction « maj » et le second thread exécute la fonction « affiche ».

2.3 Les deux threads exécutent la fonction « affiche ».

Exercice 3 (20 points) le programme suivant compile et s'exécute correctement.

```
 1  #include <iostream>
 2
 3  using namespace std;
 4
 5  template<typename T> void print(T x) { cout << x; }
 6
 7  class D {
 8        int zz;
 9  public:
10        D() : zz(42) { print('a'); }
11        ~D() { print('b'); }
12        D(const D & f) : zz(f.zz) { print('c'); }
13        D & operator=(const D & f) {
14              print('d');
15              zz = f.zz;
16              return *this;
17        }
18        D operator++(int) {
19              print('e');
20              D w(*this);
21              ++*this;
22              return w;
23        }
24        D & operator++() {
25              print('f');
26              zz += 4;
27              return *this;
28        }
29        friend ostream& operator<<(ostream& out, D& v) {
30              out << v.zz;
31              return out;
32        }
33  };
34  int main() {
35        D f1; print('-');
36        ++f1; print('-');
37        f1++; print('-');
38        D f2 = f1; print('-');
39        f2 = f1; print('-');
40        cout << f2; print('-');
41        return 0;
42  }
```

3.1 Tout en développant votre réponse, que va afficher en sortie la ligne « 35 »?

3.2 Tout en développant votre réponse, que va afficher en sortie la ligne « 36 »?

3.3 Tout en développant votre réponse, que va afficher en sortie la ligne « 37 »?

3.4 Tout en développant votre réponse, que va afficher en sortie la ligne « 38 »?

3.5 Tout en développant votre réponse, que va afficher en sortie la ligne « 39 »?

3.6 Tout en développant votre réponse, que va afficher en sortie la ligne « 40 »?

3.7 Tout en développant votre réponse, que va afficher en sortie le précédent programme (regrouper les affichages 3.1 à 3.6, puis compléter la réponse si cela s'avère nécessaire)?

Exercice 4 (20 points) le programme suivant compile et s'exécute correctement.

```
 1  #include <iostream>
 2  #include <vector>
 3  #include <algorithm>
 4  #include <functional>
 5  #include <iterator>
 6
 7  using namespace std;
 8
 9  int main() {
10        vector<int> v;
11        copy(istream_iterator<int>(cin),istream_iterator<int>(),\
12                                                back_inserter(v));
13        unique(v.begin(), v.end());
14        sort(v.begin(), v.end());
15        copy(v.begin(), v.end(),ostream_iterator<int>(cout, "\n"));
16        return 0;
17  }
```

Le programme lit des entiers de « cin » et affiche le résultat dans « cout ». Cette lecture se fait dans un ordre trié en prenant soin de retirer les doublons afin d'obtenir l'affichage désiré.

« entree.txt »	« sortie.txt »	Affichage désiré
1 2 3 4 3 2 1 8 5 2	1 1 2 2 2 3 3 4 5 8	1 2 3 4 5 8

4.1 En exécutant le programme à l'aide de la commande « `exo5.exe < entree.txt > sortie.txt` », on constate que le résultat est différent de l'affichage désiré. Il est où le bogue? Pour cette question, on ne vous demande pas d'écrire du code, vous allez le faire plus tard dans l'exercice. On vous demande juste des explications.

4.2 Nous allons modifier l'ordre des instructions « 13 » et « 14 » comme suit :

```
13        sort(v.begin(), v.end());
14        unique(v.begin(), v.end());
```

Le programme compile et s'exécute correctement. Quel sera l'affichage obtenu en sortie en utilisant la commande « `exo5.exe < entree.txt > sortie.txt` »? Est-ce qu'on obtient l'affichage désiré?

4.3 Modifiez le contenu de la fonction « main » afin d'obtenir l'affichage désiré, en utilisant que les algorithmes « copy », « erase », « sort » et « unique ».

44.4 Modifiez le contenu de la fonction « main » afin d'obtenir l'affichage désiré, en n'utilisant cette fois-ci que les algorithmes « copy », « sort », et « unique ». L'idée est de voir comment il serait possible de combiner les méthodes « copy » et « unique ».

4.5 Tout en gardant la même philosophie du programme : lecture par « cin », écriture sur « cout » et l'affichage désiré; quel serait le conteneur idéal à utiliser pour éviter de faire appel aux méthodes « sort » et « unique ». Écrivez la nouvelle fonction « main » en vous servant de ce conteneur.

Exercice 5 (20 points) dans cet exercice, il vous sera demandé de compléter le programme pour qu'il puisse fonctionner correctement.

```
 1  #include <iostream>
 2
 3  template<typename T> void print(T a) { std::cout << a; }
 4
 5  class X {
 6        int valeur;
 7  public:
 8        X(int i) : valeur(i) { print('a'); }
 9        // ajouter les choses nécessaires
10  };
11
12  // ajouter les choses nécessaires
13
14  class Y {
15        int valeur;
16  public:
17        Y(int i) : valeur(i) {}
18        // ajouter les choses nécessaires
19  };
20
21  int main() {
22        for (auto e : Y(7))
23              print(e);
24
25        std::cout << std::endl;
26  }
```

La clé de l'exercice est la ligne « 22 ». Pour rappel, cette ligne est une boucle sur l'ensemble des éléments de « Y », c'est-à-dire de « 0 » à « 6 », que l'on passe par la suite comme argument à la fonction générique « print ». L'affichage en sortie est comme suit : **aa0a1a2a3a4a5a6a**

L'affichage des deux premiers « a » est provoqué par la boucle « for ». Par la suite, les entiers de 0 à 6 sont affichés par « Y » en alternance avec la lettre « a », affichée par « X ».

Nous avons explicitement précisé dans le programme les 3 emplacements où il vous sera possible d'ajouter du code. Comme point de départ, voir quels sont les opérateurs dont vous allez avoir besoin.

Exercice 6 (20 points) soit les fichiers suivants :

<table>
<tr><th>« entree.txt »</th><th>« test.cpp.log »</th></tr>
<tr><td rowspan="3">3
test.cpp: Un programme de test
main.cpp: Le programme main
test.cpp: Une version revisee du programme test
exemple.cpp: Quelques lignes de code en exemple</td><td>test.cpp: Un programme de test
test.cpp: Une version revisee du programme test</td></tr>
<tr><td>« main.cpp.log »</td></tr>
<tr><td>Le programme main</td></tr>
</table>

Le programme est exécuté comme suit : « `exo6.exe < entree.txt` ».

Le programme va lire une série de lignes de l'entrée standard « cin ». La première ligne doit contenir un entier qui correspond au nombre de lignes à considérer. Dans notre exemple, le programme ne va tenir compte que des 3 lignes successives fournies en « cin ». Chaque ligne aura la forme « nom du fichier: texte ». Le programme va extraire le nom du fichier de chaque ligne pour créer un nouveau fichier en lui ajoutant l'extension « .log ». Dans notre exemple, il va créer les deux fichiers « test.cpp.log » et « main.cpp.log ». Pour chacun des fichiers, il va regrouper les textes correspondants.

Écrire le programme permettant de réaliser une telle tâche. Comme le code n'est pas volumineux, il vous est possible de l'écrire directement dans la fonction « main ». **Attention, pas d'usine à gaz!**

Lors de la correction, je vais tenir compte de la clarté de votre code et de son optimisation.

Joyeuses Fêtes!