

Trimestre Automne, 2017

Mohamed Lokbani

IFT1169 – Examen Intra –

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

Nom : _____ Prénom(s) : _____

Signature : _____ Login : _____

Date : Vendredi 20 octobre 2017

Durée : 2 heures (de 16h30 à 18h30)

Local : Z-310, Pavillon Claire McNicoll

Directives :

- Toute documentation est permise.
- Appareils électroniques **non** permis.
- Répondre directement sur le questionnaire.
- Les réponses **doivent être brèves, précises, claires et nettement présentées.**

1. _____ /20 (1.1)
2. _____ /10 (2.1, 2.2, 2.3, 2.4, 2.5)
3. _____ /20 (3.1)
4. _____ /15 (4.1)
5. _____ /15 (5.1)
6. _____ /20 (6.1)

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 (20 points) Trouvez la meilleure correspondance pour les descriptions ci-dessous en y associant un nombre de 1 à 11. Le nombre ne peut-être utilisé qu'une seule fois. Nous avons complété une description à titre d'exemple.

Une ____template____ (1) n'est pas ____ (2) qu'il suffit d'appeler, c'est ____ (3) qui va générer une fonction normale lors de ____ (4).

Schématiquement, l'instanciation d'un template se compose de trois phases :

on ____ (5) dans le template les différents arguments template par leur vraie valeur (ce qui peut déclencher l'instanciation d'autres templates) ;

on ____ (6) le code résultant ;

on peut enfin ____ (7) à la fonction générée.

Les privilèges de l'amitié ne sont pas ____ (8). Si la classe Fred déclare que la classe Wilma est une amie, et que la classe Wilma déclare que Betty est une amie, la classe Betty n'a pas à avoir automatiquement des droits d'accès particuliers aux objets de type ____ (9).

Les privilèges de l'amitié ne sont pas ____ (10). Si la classe Fred déclare que la classe Wilma est une amie, les objets de type Fred n'ont pas à avoir automatiquement des droits d'accès particuliers aux objets de type ____ (11).

compile	
Fred	
remplace	
template	1
l'instanciation	
Wilma	
se lier	
une fonction normale	
réciroques	
un modèle	
transitifs	

Exercice 2 (10 points) Soit une pile générique du type « LIFO » comme celle étudiée en cours. Parmi les méthodes définies dans la pile, nous avons :

```
bool isEmpty( );  
void push(int item);  
int pop( );  
int top( );
```

Soit le fragment de code suivant :

```
IntStack s; // une pile d'entiers  
  
int n1, n2, n3;  
  
s.push(17);  
s.push(143);  
s.push(42);  
  
n1 = s.pop( );  
n2 = s.top( );  
s.push(n1);  
n3 = s.pop( );  
n1 = s.top( );
```

Après exécution du programme.

2.1 Quelle est la valeur au « top » de la pile « s »? (une courte explication)

2.2 Quelle est la valeur au «fond » de la pile « s »? (une courte explication)

2.3 Quelle est la valeur de « n1 »? (une courte explication)

2.4 Quelle est la valeur de « n2 »? (une courte explication)

2.5 Quelle est la valeur de « n3 »? (une courte explication)

Exercice 3 (20 points) La classe générique « Pair » est une structure contenant deux éléments éventuellement de types différents. On vous demande de compléter et/ou de corriger le programme ci-dessous afin de pouvoir exécuter la fonction « main » suivante:

```
int main() {
    Pair<int,int> p1(1,4), p2(1,6), p3(2,1);
    operator<(p1,p2)? (cout << p1<<endl): (cout <<p2<<endl); // (1)
    operator<(p2,p3)? (cout << p2<<endl): (cout <<p3<<endl); // (2)
    return 0;
}
```

- (1) affiche (1,4) c'est-à-dire la paire associée à p1.
(2) affiche (1,6) c'est-à-dire la paire associée à p2.

Le programme est :

```
1  #include <iostream>
2
3  using namespace std;
4
5  template <class T1, class T2> class Pair
6  public:
7      Pair( T1 x, T2 y ) : membreFirst( x ), membreSecond( y ) {}
8      T1 first() { return membreFirst; }
9      T2 second() { return membreSecond; }
10 private:
11     T1 membreFirst; T2 membreSecond;
12 };
13
14 // Doit ordonner par first puis second, par exemple (1,4) < (1,6) < (2,1)
15
16 bool operator<( const Pair<T1, T2> &p1, const Pair<T1, T2> &p2 ) {
17     return p1.first() < p2.first();
18 }
19
20
21 void operator<<( ostream out, const Pair<T1, T2> &p ) {
22     out << "(" << p.first() << "," << p.second() << ")";
23 }
24
```

La réponse doit-être sous la forme : numéro de la ligne, en mentionnant s'il s'agit d'une modification ou d'un ajout, en y incluant à chaque fois une courte explication.

Exercice 4 (15 points) Soit la classe générique « unobj ». On vous demande de compléter la méthode « unobj operator+(unobj &m) »

```
1 template<class T>
2 class unobj {
3 public:
4     unobj(T x): val(x) {}
5     unobj operator+(unobj &m) {
6         // méthode à compléter
7     }
8 private:
9     T val;
10 };
```

Exemple d'utilisation de la classe « unobj » pour des entiers :

```
unobj<int> p1(5); // p1 : val=5
unobj<int> p2(7); // p2 : val=7
unobj<int> p3 = p1+p2; // p3 : val=12
```

Exercice 5 (15 points) En utilisant autant que possible les conteneurs de la STL et les algorithmes, écrivez le code de la fonction « void print_unique(int a[], int taille_a) » qui permet d’afficher en sortie, dans un ordre décroissant, les nombres uniques d’un tableau. La fonction ne modifie pas le contenu du tableau « a ».

Un exemple d’utilisation de la méthode « print_unique_inverse » :

```
int main(){
    int a[] = { 8, 4, 1, 2, 4, 3, 1, 5, 2 };
    print_unique(a,9);

    return 0;
}
```

Ce programme va afficher en sortie :

Items uniques = 8 5 4 3 2 1

Exercice 6 (20 points) En utilisant autant que possible les conteneurs de la STL et les algorithmes, écrivez le code de la fonction « `print_plus_frequent(int a[], taille_a)` » qui retourne l'élément le plus fréquent dans un tableau. S'il existe plusieurs nombres, elle retourne le plus petit.

Un exemple d'utilisation de la méthode « `get_plus_frequent` » :

```
int main() {
    int a[] = {6, 5, 4, 2, 6, 5, 8, 2, 8, 3, 2, 8};
    cout << print_plus_frequent(a,12) << endl;

    return 0;
}
```

Ce programme va afficher en sortie :

2

}