Contrôle Continu LO43 – lundi 18 mai 2015
(durée 2h - aucun document autorisé)

I - Considérons le type abstrait vecteur en C++. On s'intéresse à l'opérateur d'incrémentation postfixe v++ en comparaison avec celui préfixe ++v qui tous les deux incrémentent les composantes du vecteur d'1 unité avec la différence que l'opérateur postfixe ne renvoie pas le vecteur incrémenté mais la valeur du vecteur avant l'incrémentation. Noter que le paramètre *int* permet de distinguer leur écriture :

```
... operator++(void) { // préfixe ...}
... operator++(int i) { // postfixe ...}
```

a) Donner les contenus des vecteurs r1 et r2 et des vecteurs u et v après exécution de :

```
vecteur u("2 2 2"), v("2 2 2"); vecteur r1, r2;
r1 = u++; r2 = ++v;
```

Donner également les 2 dernières instructions en notation fonctionnelle équivalente.

b) Ecrivez les deux opérateurs complètement.

II - On dispose d'une classe Liste dont les éléments sont des instances de la classe T. On veut définir une classe File de type FIFO (first in/first out) à partir de la classe Liste. Les opérations pour cette file sont "ajouter", "supprimer" et "tete". L'interface de la classe Liste est telle que vue en TD et/ou TP.

a) Rappeler quelles sont les deux méthodes permettant de faire de la réutilisation.
b) Donner l'implémentation de la classe File suivant chacune de ces 2 méthodes.

III - Soit une classe Animal dont on n'impose pour l'exercice que le code suivant :

```
class Animal { String nom ; public: ...  } ;

main() {
  Animal* ptr[3] ;
  ptr[0] = new Chat("miaou") ;
  ptr[1] = new Oiseau("pioupiou") ;
  ptr[2] = new Chat("minou") ;
  for (int i=0;i<3;i++) {
    ptr[i]->affiche() ;
  }
}
```

a) Définir les classes (interface et implantation) Animal, Chat et Oiseau pour que le résultat du main soit le suivant :

```
Chat miaou
Oiseau pioupiou
Chat minou
```

b) A votre avis, la classe Animal doit-elle ici vérifier la forme de Coplien ? Pourquoi ?

c) Réécrire le main dans le cas où on utilise un tableau de valeurs et non pas de pointeurs ?

```
main() { Animal ptr[3]; ... }
```

IV - Indiquer quelles sont les erreurs de syntaxe ou de logique présentes dans les programmes suivants, s'il y en a, et qui devraient être renvoyées par le compilateur.

a)
```
class A {}; class B : public A { }; A a; B b; b = a;
```

b)
```
class B {protected:  int i; friend void f();};
class D : public B {};
void f( ) {B* p = new B; D* q = new D; int fi1 = p->i; int fi2
= q->i;}
```

c)
```
class B {protected:  int i;};
class D : public B {  };
void f( ) {B* p = new B; D* q = new D; int fi1 = p->i; int fi2
= q->i;}
```

d)
```
class A {protected: int a;};
class B : public A {void f( );};
class C : private B {void f();};
void B::f( ) {a = 1;}
void C::f( ) {a = 1;}
```

e)
```
class A {protected: int a;};
class B : private A {void f( );};
class C : public B {void f( );};
void B::f( ) {a = 1;}
void C::f( ) {a = 1;}
```

f)
```
class A {protected: int a;};
class B : protected A {void f( );};
class C : private B {void f( );};
void B::f( ) {a = 1;}
void C::f( ) {a = 1;}
```