Trimestre Été, 2006

Mohamed Lokbani

IFT1169 – Examen Intra –

Inscrivez tout de suite votre nom et code permanent.

Nom: _____| Prénom(s): _____|

Signature: _____| Code perm: _____|

Date : mercredi 31 mai 2006

Durée : 2 heures (de 17h30 à 19h30)

Local : Z-110 ; Pavillon Claire-McNicoll (ancienne aile Z).

Directives:

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

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

1._____/30 (1.1, 1.2, 1.3, 1.4, 1.5, 1.6)

2._____/20 (2.1, 2.2, 2.3a, b, c, d)

3._____/25 (3.1)

4._____/25 (4.1)

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 (30 points)

1.1 Expliquer pourquoi l'instruction (#1) est valide (correcte) mais pas l'instruction (#2) :

```
int tab[10],*ptr1,*ptr2;
ptr1 = tab ;  // instruction valide -#1-
tab  = ptr2 ; // instruction invalide -#2-
```

1.2 Quand est-il nécessaire de fournir à votre classe un constructeur de recopie explicit et/ou un destructeur explicit afin d'éviter d'utiliser ceux fournis par défaut ? Aussi, quand faut-il surcharger les opérateurs « != », « == » et/ou « = » de votre classe, au lieu d'utiliser ceux fournis par défaut ? Expliquer vos réponses.

1.3 Les méthodes statiques ne peuvent pas être déclarées virtuelles (« virtual »). Elles ne sont donc pas polymorphiques!

Vrai Faux Explications :

1.4 Donner une distinction importante dans l'utilisation de « typeid » vis-à-vis de « dynamic_cast » dans le cadre d'une hiérarchie des classes.

1.5 Soit la déclaration suivante:

```
class A {public: int x; protected: int y;};
```

Laquelle des déclarations ci-dessous sera rejetée par le compilateur?

```
a)     class B: public A { void f() { x = y; }};
b)     class B { void f() { A a; a.x = a.y; }};
c)     class B: public A{}; class C: public B { void f() { x = y; }};
d)     Aucune des précédentes réponses.
```

No : Explications :

1.6 Soit la classe « A » définie comme suit :

```
class A {
public:
    int operator()() { return 5; }
    A operator+(int) { return *this; }
};
```

Laquelle des déclarations ci-dessous sera rejetée par le compilateur?

```
a)     A f; int x = f();
b)     A x,y; x = x+y;
c)     A x; x = x+3;
d)     A f,x; f = f+x();
```

No : Explications :

Exercice 2 (20 points) Soit le fragment de code suivant :

```
 1    #include <iostream>
 2    #include <cassert>
 3    using namespace std;
 4
 5    namespace {
 6        int add(int i, int j) { return i+j;}
 7    }
 8
 9    int sous(int i, int j) { return i-j;}
10
11    namespace N1 {
12        int add(int i, int j) { return i+j;}
13        int mult(int i, int j) { return i*j;}
14        int sous(int i, int j) { return i-j;}
15        int div(int i, int j) { assert(!j); return i/j;}
16    }
17    // une fonction main …
```

2.1 Nommer les fonctions définies dans l'espace anonyme.

2.2 Nommer les fonctions définies dans l'espace global.

2.3 Pour chacune des méthodes « main », dire si elle est correcte ou incorrecte. Dans le cas où elle est correcte, dire aussi ce qu'elle va afficher en sortie.

a)

```
1    int main() {
2        cout << sous(2,1) << " " << add(2,1)<<endl;
3        return 0;
4    }
```

Correcte ☐ Incorrecte ☐ Affichage ☐

Explications :

b)

```
1   int main() {
2       using namespace N1 ;
3       cout << sous(2,1) << " " << mult(2,1) << endl;
4       return 0
5   }
```

Correcte ☐ Incorrecte ☐ Affichage ☐

Explications :

c)

```
1   int main() {
2       using N1::div;
3       cout << sous(2,1) << " " << div(2,1) << endl;
4       return 0 ;
5   }
```

Correcte ☐ Incorrecte ☐ Affichage ☐

Explications :

d)

```
1   int main() {
2       cout << sous(2,1) << " " << N1::sous(2,1) << endl;
3       return 0 ;
4   }
```

Correcte ☐ Incorrecte ☐ Affichage ☐

Explications :

Exercice 3 (25 points) Compléter le programme suivant :

```
01  #include <iostream>
02  #include <string>
03  using namespace std;
04
05  // Insérer votre code ici
06
07  int main() {
08      Nourriture fruit1("banane", 65); // Nom (défaut : « pas de nom »),
09                                       // Calories (défaut: 100)
10      fruit1.affiche();
11      cout << endl;
12      Nourriture fruit2;
13      fruit2.setNom("pomme") ;
14      fruit2.affiche();
15      cout << endl;
16      Nourriture salade = fruit1 + fruit2;
17      salade.setNom("Salade de fruits");
18      salade.affiche();
19      cout << endl;
20      return 0;
21  } // Fin main
```

L'affichage en sortie est comme suit :

```
banane (65 calories/100g)
pomme (100 calories/100g)
Salade de fruits (82 calories/100g)
```

Exercice 4 (25 points) Que va afficher en sortie le programme suivant qui compile et s'exécute correctement.

```
01  #include <iostream>
02  using namespace std;
03  class TropPetitException {
04  public:
05      TropPetitException(char* tse){}
06      char* affiche() {return "TropPetit";}
07  };
08  class TropGrandException {
09  public:
10      TropGrandException(char* tse){}
11      char* affiche() {return "TropGrand";}
12  };
13  class PairException {
14  public:
15      PairException(char* tse){}
16      char* affiche() {return "Pair";}
17  };
18  void fonc1 ( int x ) throw(TropPetitException,TropGrandException){
19      cout << x << endl;
20      cout << "f1-1" << endl;
21      if ( x < 0 ) {
22          throw TropPetitException("Trop petit");
23      }
24      cout << "f1-2" << endl;
25      if ( x > 1 ) {
26          throw TropGrandException("Trop grand");
27      }
28      cout << "f1-3" << endl;
29  }
30  void fonc2 ( int z ) throw(TropGrandException,PairException){
31      cout << z << endl;
32      cout << "f2-4" << endl;
33      try {
34          fonc1(z);
35          cout << "f2-5" << endl;
36      } catch ( TropPetitException e ) {
37          cout << "f2-6" << endl;
38          if ( z % 2 == 0 ) {
39              throw PairException("Pair");
40          }
41          cout << "f2-7" << endl;
42      }
43      cout << "f2-8" << endl;
44  }
45  int main (){
46      cout << "m-9" << endl;
47      try {
48          for ( int ctr = -2 ; ctr < 5 ; ctr++ ) {
49              cout << "*" << ctr << endl;
50              try {
51                  fonc2(ctr);
52                  cout << "m-10" << endl;
53                  ctr = -ctr;
54              } catch ( TropGrandException e1 ) {
55                  cout << e1.affiche() << endl;
56                  fonc1(-ctr);
57                  cout << "m-11" << endl;
58              } catch ( PairException e2 ) {
59                  cout << e2.affiche() << endl;
60              }
61              cout << "m-12" << endl;
62          }
63      } catch ( TropPetitException e3 ) {
64          cout << e3.affiche() << endl;
65      }
66      cout << "m-13" << endl;
67      return 0;
68  }
```

Affichage en sortie (avec explications) :