Trimestre Hiver, 2012

Mohamed Lokbani

IFT1169 – Examen Intra –

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

Nom : _____| Prénom(s) : _____|

Signature : _____| Code perm : _____|

Date : mardi 21 février 2012

Durée : 2 heures (de 18h30 à 20h30)

Local : Z-210 ; 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._____/30 (1.1. à 1.7)

2._____/20 (2.1)

3._____/20 (3.1 3.2)

4._____/20 (4.1)

5._____/10 (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 (30 points)

1.1 Citer le nom d'un langage de programmation orientée objet.

1.2 Associer les lettres aux descriptions (une lettre par description).

A. Classe dérivée
B. Fonction virtuelle
C. Dissimulation d'informations
D. Héritage
E. Redéfinition des méthodes
F. Classe de base
G. Ligature statique
H. Surcharge des opérateurs
I. Ligature dynamique
J. Fonction amie
K. Polymorphisme

La possibilité d'avoir un même appel de fonction produisant des réponses différentes en fonction de la nature de l'instance qui fait l'appel.

Une fonction est utilisée quand il faut implanter la ligature dynamique.

La possibilité de dériver une classe d'une autre classe.

Le principe sur lequel une classe est construite n'est pas important à un programmeur qui veut utiliser ladite classe.

Une nouvelle classe est créée à partir d'une classe existante.

1.3 Le nom d'un tableau peut très bien être utilisé comme un pointeur sur son premier élément.

1.4 Que signifient ces 2 déclarations :

```
int (*a)[100];
```

```
int *a[100];
```

1.5 La fonction `rand` n'accepte aucun argument et retourne un entier, généré (pseudo) aléatoirement dans un intervalle 0 et `RAND_MAX`.

Si X & Y sont deux nombres du type « double », parmi les 3 instructions ci-dessous, laquelle permet de générer un nombre aléatoire, du type double, compris entre X & Y. Expliquer brièvement votre réponse.

R1	`Z = X + (X-Y) * static_cast<double> (rand())/RAND_MAX;`
R2	`Z = rand()%X + Y;`
R3	`Z = X + (Y-X) * static_cast<double> (rand())/RAND_MAX;`

1.6 Si X est une instance d'une classe donnée, les membres de cette classe sont accessibles à travers l'opérateur _____. Si Y est un pointeur vers l'instance X, les membres de cette classe sont accessibles (via Y) à travers l'opérateur _____.

1.7 Tout en expliquant brièvement votre réponse, que va afficher en sortie le programme suivant qui compile et s'exécute correctement?

```
#include <iostream>

using namespace std;

int main(){
        int *p1, *p2, x=6;
        p1 = new int;
        *p1 = x*20;
        p2 = p1;
        cout << "A: " << x * *p1 << " " << *p2 << endl;
        *p2 = 30;
        cout << "B: " << x * *p1 << " " << *p2 << endl;
        p1 = new int;
        *p1 = 40;
        cout << "C: " << x * *p1 << " " << *p2 << endl;

        return 0;
}
```

.

Exercice 2 (20 points)

Expliquer l'utilité du programme ci-dessous qui compile et s'exécute correctement. Donner l'affichage en sortie si nous fournissons au programme les deux chaînes « test » et « rien ».

```
#include <iostream>

using namespace std;

void mystere(char* s1, const char* s2){
        while (*s1 != '\0')
                ++s1;

        for ( ; (*s1 = *s2); s1++, s2++)
                ;
}

int main(){
        char chaine1[80], chaine2[80];

        cout << "Entrer 2 chaines de caractères < 80\n";
        cin >> chaine1 >> chaine2;
        mystere(chaine1, chaine2);
        cout << chaine1 << endl;
        cout << chaine2 << endl;

        return 0;
}
```

Exercice 3 (20 points)

Soit le programme suivant :

```
#include <iostream>

using namespace std;

class base{
        int haut;
        int bas;
public:
        base(); // haut=0, bas=1
        base(int zz); // haut=zz, bas=1
        void PrintHaut() { cout << "base Haut = " << haut << endl;}
        void PrintBas() { cout << "base Bas = " << bas << endl;}
};

class derivee: public base {
        int haut;
        int bas;
public:
        derivee(); // base(), haut=3, bas=4
        derivee(int zz); // base(zz+1), haut=zz, bas=3
        void PrintHaut() { cout << "derivee Haut = " << haut << endl;}
        void PrintBas() { cout << "derivee Bas = " << bas << endl;}
        void PrintBaseHaut() { base::PrintHaut();}
        void PrintBaseBas() { base::PrintBas();}
};
```

3.1 Écrire le code complet des constructeurs de la classe de base et de la classe dérivée en tenant compte des valeurs par défaut définies dans les commentaires du programme.

3.2 Que va afficher en sortie le précédent programme ayant la fonction « main » ci-dessous :

```
int main(){
        derivee z1;
        derivee z2(6);

        z1.PrintHaut();
        z1.PrintBas();
        z1.PrintBaseHaut();
        z1.PrintBaseBas();

        z2.PrintHaut();
        z2.PrintBas();
        z2.PrintBaseHaut();
        z2.PrintBaseBas();

        return 0;
}
```

Exercice 4 (20 points)

Le programme suivant compile et s'exécute correctement :

```
#include <iostream>
using namespace std;

class XBase {
};
class XDerivee1: public XBase {
};
class XDerivee2: public XDerivee1 {
};
class YBase {
public:
        virtual void test() { }
        virtual void test(double v) { }
        virtual void test(string s) { }
        virtual void test(const XBase * xo) { }
        virtual void test(const XDerivee2 * xd2) { }
};
class YDerivee1: public YBase {
public:
        virtual void test(int v) { }
        virtual void test(const XBase * xo) { }
        virtual void test(const XDerivee1 * xd1) { }
};
class YDerivee2: public YDerivee1 {
public:
        virtual void test(const XBase * xo) { }
        virtual void test(const XDerivee1 * xd1) { }
        virtual void test(const XDerivee2 * xd2) { }
};
int main(int argc, char * argv[]) {
        XBase * xbb = new XBase;
        XBase * xbd2 = new XDerivee2;
        XDerivee1 * xd1d2 = new XDerivee2;
        YBase * ybd1 = new YDerivee1;
        YBase * ybd2 = new YDerivee2;
        YDerivee1 * yd1d2 = new YDerivee2;
        ybd1 -> test(xbb);
        ybd1 -> test(xbd2);
        ybd2 -> test(xbd2);
        ybd2 -> test(xd1d2);
        yd1d2 -> test(xd1d2);
        delete xbb;
        delete xbd2;
        delete xd1d2;
        delete ybd1;
        delete ybd2;
        delete yd1d2;

        return 0;
}
```

Tout en expliquant brièvement votre réponse, citer l'ensemble des appels à la méthode « test » en précisant à quelle classe elle appartient, et quelle est la signature de la méthode « test » dans la classe en question?

Exercice 5 (10 points)

Soit la classe « Vecteur » :

```
class Vecteur {
private:
        double v_x;
        double v_y;
        double v_z;
public:
        Vecteur(double x, double y, double z) { v_x = x; v_y = y; v_z = z; }
};
```

Écrire la surcharge des 3 opérateurs suivants pour la classe « Vecteur » : l'opérateur négation (`-v`), la surcharge de l'opérateur de sortie (`<<v`) et finalement l'opérateur Préfix (`++v`). Si « v » est une instance de « Vecteur », ces opérateurs auront un effet sur les 3 composantes « v_x, v_y et v_z » de l'instance « v ».