Trimestre Hiver, 2009

Mohamed Lokbani

IFT1169 – Examen Intra –

Inscrivez tout de suite votre nom et le code permanent.

Nom: _____| Prénom(s): _____|

Signature: _____| Code perm: _____|

Date : mardi 24 février 2009

Durée : 2 heures (de 18h30 à 20h30)

Local : Z-220 ; 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._____/10 (1.1)

2._____/18 (2.1, 2.2, 2.3, 2.4, 2.5, 2.6)

3._____/18 (3.1)

4._____/24 (4.1)

5._____/30 (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 (10 points)

Pour chaque terme lui associer une seule lettre des descriptions ci-dessous, qui lui correspond le mieux.

	Terme	Correspondance
1	?:	
2	Surcharge des opérateurs	
3	L'opérateur d'accès à un membre de la classe	
4	Constructeur de recopie	
5	Variable statique de la classe	
6	Affectation membre à membre	
7	Destructeur	
8	new et delete	
9	Fonction amie	
10	L'encapsulation	

Descriptions

A) Un constructeur qui prend comme argument un objet de la même classe que la classe courante.

B) C'est le comportement par défaut de l'opérateur « = ».

C) Le processus qui permet aux opérateurs du langage C++ d'avoir des classes objets comme opérandes.

D) Un opérateur du langage C++ qui ne peut être surchargé.

E) Même si défini(e) à l'extérieur de la classe, a quand même accès aux membres privés de la classe.

F) Utilisé uniquement quand une copie de la variable doit être partagée par toutes les instances de la classe.

G) Opérateurs utilisées pour l'allocation et la libération de la mémoire.

H) L'inclusion d'une chose dans une autre de telle manière qu'elle ne soit pas apparente.

I) L'opérateur « . » ou « -> ».

J) Il a la tâche de faire le ménage dans « la maison ».

Exercice 2 (18 points) Supposer que la classe générique à l'entête suivante:

```
template <class T1> class C1;
```

Décrire les relations d'amitié établies lorsqu'on place chacune des déclarations ci-dessous, respectivement, dans la classe générique « C1 ». Les identifiants commençant par la lettre « f » correspondent à des fonctions, par la lettre « T » à un type quelconque et par la lettre « C » à des classes.

2.1 friend void f1();

2.2 friend void f2(C1 < T1 > &);

2.3 friend void C2 :: f4();

2.4 friend void C3 < T1 > :: f5(C1 < T1 > &);

2.5 friend class C5;

2.6 friend class C6 < T1 >;

Exercice 3 (18 points) Introduire les changements nécessaires pour que la classe « Paire », ci-dessous, puisse être une classe générique dont les champs membres « x » et « y », seront de **types différents et** qui supportent l'opération +.

```
01  class Paire {
02
03          int x, y;
04
05  public:
06
07          Paire (int a, int b): x(a), y(b) { }
08
09          int getX ( ) { return x; }
10
11          int getY ( ) { return y; }
12
13          void setX (int a) { x=a; }
14
15          void setY (int b) { y=b; }
16
17          Paire & operator+ (const Paire & p) const {
18
19                  return Paire(x+p.x, y+p.y);
20          }
21  };
22
```

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
04  class Voiture {
05
06        int vitesse;
07  public:
08        Voiture( ) { vitesse=0; }
09        int getVitesse( ) { return vitesse; }
10        void acceleration1(int delta) { vitesse+=delta; }
11        virtual void acceleration2(int delta) { vitesse+=delta; }
12  };
13
14  class VoitureCourse: public Voiture {
15
16  public:
17        VoitureCourse( ): Voiture( ) { }
18        void acceleration1(int delta) {
19              Voiture::acceleration1(2*delta);
20        }
21        virtual void acceleration2(int delta) {
22              Voiture::acceleration2(2*delta);
23        }
24  };
25
26  void stop1(Voiture& x) { x.acceleration1(-x.getVitesse( )); }
27
28  void stop2(Voiture& x) { x.acceleration2(-x.getVitesse( )); }
29
30  int main( ) {
31
32        Voiture c1, c2;
33        c1.acceleration1(65);
34        cout << c1.getVitesse( ) << endl;
35  // Affichage -1- avec une courte explication
36
37
38
39
40        stop1(c1);
41        cout << c1.getVitesse( ) << endl;
42  // Affichage -2- avec une courte explication
43
44
45
46
47        c2.acceleration2(65);
48        cout << c2.getVitesse( ) << endl;
49  // Affichage -3- avec une courte explication
50
51
52
53
54        stop2(c2);
55        cout << c2.getVitesse( ) << endl;
56  // Affichage -4- avec une courte explication
57
58
59
60
61        VoitureCourse r1, r2;
62        r1.acceleration1(100);
63        cout << r1.getVitesse( ) << endl;
64  // Affichage -5- avec une courte explication
65
66
67
68
```

```
69          stop1(r1);
70          cout << r1.getVitesse( ) << endl;
71  // Affichage -6- avec une courte explication
72
73
74
75
76          r2.acceleration2(100);
77          cout << r2.getVitesse( ) << endl;
78  // Affichage -7- avec une courte explication
79
80
81
82
83          stop2(r2);
84          cout << r2.getVitesse( ) << endl;
85  // Affichage -8- avec une courte explication
86
87
88
89
90          Voiture *p1=new VoitureCourse, *p2=new VoitureCourse;
91          p1->acceleration1(100);
92          cout << p1->getVitesse( ) << endl;
93  // Affichage -9- avec une courte explication
94
95
96
97
98          stop1(*p1);
99          cout << p1->getVitesse( ) << endl;
100 // Affichage -10- avec une courte explication
101
102
103
104
105         p2->acceleration2(100);
106         cout << p2->getVitesse( ) << endl;
107 // Affichage -11- avec une courte explication
108
109
110
111
112         stop2(*p2);
113         cout << p2->getVitesse( ) << endl;
114 // Affichage -12- avec une courte explication
115
116
117
118
119         return 0;
120 }
```

Exercice 5 (30 points) Le programme suivant compile et s'exécute correctement.

```
01  #include <iostream>
02  #include <string>
03  using namespace std;
04
05  const int EmployeSalarie=1, EmployeHeure=2, EmployeCommission=3;
06
07  struct Employe {
08     string nom;
09     int EmployeType;
10     float SalaireAnnuel;
11     float TauxHoraire;
12     float Commission;
13  };
14
15  float CalculPayeSemaine (Employe &e, float HeuresTravaillees,
16                                                  float VentesTotales) {
17     double x = 0.0;
18     switch (e.EmployeType) {
19          case EmployeSalarie:
20                 return e.SalaireAnnuel / 52; break;
21          case EmployeHeure:
22                 return HeuresTravaillees * e.TauxHoraire; break;
23          case EmployeCommission:
24                 return VentesTotales * e.Commission/100; break;
25     }
26     return x;
27  }
28
29  int main( ) {
30     struct Employe e[3];
31     e[0].nom="Alain";
32     e[0].EmployeType=EmployeSalarie;
33     e[0].SalaireAnnuel=52000.00;
34     cout << e[0].nom << ':' << endl;
35     cout << '$' << CalculPayeSemaine (e[0], 40, 0) << endl << endl;
36     e[1].nom="Bertrand";
37     e[1].EmployeType=EmployeHeure;
38     e[1].TauxHoraire=17.50;
39     cout << e[1].nom << ':' << endl;
40     cout << '$' << CalculPayeSemaine (e[1], 50, 8000) << endl << endl;
41     e[2].nom="Michel";
42     e[2].EmployeType=EmployeCommission;
43     e[2].Commission=5.00;
44     cout << e[2].nom << ':' << endl;
45     cout << '$' << CalculPayeSemaine (e[2], 10, 25000) << endl << endl;
46     return 0;
47   }
```

L'affiche en sortie obtenu après l'exécution du programme :

Alain:
$1000

Bertrand:
$875

Michel:
$1250

Question :

Il vous est demandé de réécrire le programme en utilisant un bon style orienté objet, en y incluant des membres privés, constructeurs nécessaires, l'héritage, ligature dynamique, et des variables polymorphiques. Éliminer du programme toutes les constantes, « struct », l'instruction de contrôle « switch/case », code redondant, les fonctions ordinaires (sauf la fonction « main »). Mise à part les modifications introduites pour satisfaire le style orientée objet, votre programme doit être équivalent à l'original et doit afficher les mêmes résultats en sortie.