UQÀM **Université du Québec à Montréal**

INF-3105, Structures de données et algorithmes
— 2009

Examen intra
— 2009

CONSIGNES

- **Les règlements de l'UQAM concernant le plagiat seront strictement appliqués.**
- Aucune sortie n'est permise durant l'examen.
- Il est important de bien expliquer vos choix s'il y a lieu.
- Aucun document n'est permis.
- La durée de l'examen est de 3 heures.
- Vous pouvez utiliser les versos comme brouillon ou comme espace supplémentaire.
- **Il est interdit de dégrafer le questionnaire.**
- Les téléphones cellulaires, calculatrices, ordinateurs, palm, baladeurs, iPods, etc. sont interdits.

IDENTIFICATION

NOM : _____

PRÉNOM : _____

CODE PERMANENT : _____

SIGNATURE : _____

GROUPE : _____

PROFESSEUR : _____

#1	_____ / 0
#2	_____ / 0
#3	_____ / 0
#4	_____ / 0
#5	_____ / 0
#6	_____ / 0
#7	_____ / 0
TOTAL	_____ / 0

commentaire :

Numéro 1. (0 pts)
Objectif(s) :

- **Application des connaissances.**
- **Constructeur, Destructeur.**

Question : Que va afficher le programme suivant :

```
#include <iostream>

using namespace std;

class G {
 public :
  G(){ cout << "G" << endl; };
  virtual ~G(){ cout << "-G" << endl; };
};

class H : public G {
 public :
  H(): G() { cout << "H" << endl; };
  virtual ~H(){ cout << "-H" << endl; };
};

class K {
 public :
  K(){ cout << "K" << endl; };
  virtual ~K(){ cout << "-K" << endl; };
};

class F : public H, public K {
 public :
  F(): K(), H() { cout << "F" << endl; };
  virtual ~F(){ cout << "-F" << endl; };
};

void t() {
  F f;
}

int main() {
  t();
  return 0;
}
```

Réponse :

G
H
K
F
-F
-K
-H
-G

Numéro 2. (0 pts)
Objectif(s) :

- **Application des connaissances.**
- **Surcharge d'opérateur.**

Question : Soit la déclaration suivante :

```
class NombreComplexe
  private :
    double reelle;
    double imaginaire;
  public :
    NombreComplexe( double a_reelle = 0.0, double a_imaginaire = 0.0 ) :
      reelle( a_reelle ), imaginaire( a_imaginaire ) ;
    virtual ~NombreComplexe();
;
```

Écrivez le code qui permet de surcharger l'opérateur de multiplication (*) afin de multiplier deux nombres complexes sachant que re = x1.re * x2.re - x1.im * x2.im et im = x1.re * x2.im + x1.im * x2.re.
Réponse :

```
  NombreComplexe operator*( const NombreComplexe & v2 ) {
    NombreComplexe resultat;
    resultat.reelle = reelle * v2.reelle - imaginaire * v2.imaginaire;
    resultat.imaginaire = reelle * v2.imaginaire + imaginaire * v2.reelle;
    return resultat;
  }
```

Numéro 3. (0 pts)
Objectif(s) :

- **Synthèse de la matière.**
- **Fonction amie.**

Question : Expliquez le role du mot **friend** lorsqu'il est appliqué à une foncion.
Réponse :

Cela donne accès au différents champs de la classe. Cette accès est donnée à la fonction friend. L'accès s'applique au champs privés, protégés et publiques.

Numéro 4. (0 pts)
Objectif(s) :

- **Application des connaissances.**
- **Utilisation des pointeurs.**

Question : Dites ce que va afficher le code suivant :

```
#include <iostream>

using namespace std;

class C {
public :
  C(int a_a = 0) : a(a_a) {}
  C(const C & a_c) : a( a_c.a ) {}
  virtual ~C(){}
  C * f(){
    C * r = this;
    r++;
    return r;
  }
  friend ostream & operator<<( ostream &, C & );
private :
  int a;
};

ostream & operator<<( ostream & a_cout, C & a_c ){
  a_cout << a_c.a;
  return a_cout;
}

int main() {
  C t[10];
  int i = 0;
  for( i = 0; i < 10; i++ ){
    t[i] = C( 10 - i );
  }

  C * k = t + 3;
  cout << *k;
  *( k + 1 ) = *( k - 1 );
  cout << k[1];
  k = k->f();
  cout << k[0];

  return 0;
}
```

Réponse :

788

Numéro 5. (0 pts)
Objectif(s) :

- **Synthèse de la matière.**
- **Classe Abstraite.**

a) (0 pts) Expliquez ce qu'est une méthode abstraite pure.
Réponse :

C'est une méthode qui n'a pas de code. Elle n'est pas implémentée par la classe.

b) (0 pts) Expliquez ce qu'est une classe abstraite.
Réponse :

C'est une classe qui contient au moins une méthose abstraite. Il est impossible d'instantier cette classe.

c) (0 pts) Quel est l'utilité d'avoir des classes abstraite pure ?
Réponse :

Cela permet de construire des interfaces. Une interface nous permet de garantir des caractéristiques fonctionnelles pour une classe.

Numéro 6. (0 pts)
Objectif(s) :

- **Application des connaissances.**
- **Programmation C++.**

Question : Le code suivant contient deux (2) classes. Complétez le en ajoutant le code pour les constructeurs par défaut et le code pour les constructeurs de copies.
Note: L'entête des constructeurs par défaut est déjà inclue.

```
class Forme2D {
 public:
  Forme2D (double x = 0, double y = 0);
  virtual double surface () = 0;
 protected:
  double centre_x;
  double centre_y;
};

class Cercle : virtual public Forme2D {
 public:
  Cercle (double x = 0, double y = 0, double r = 0);
  double surface ();
 protected:
  double rayon;
};
```

Réponse :

```
Forme2D::Forme2D( double x = 0, double y = 0 ) :
  centre_x( x ), centre_y( y )
{}

Forme2D::Forme2D( const Forme2D & a_forme ) :
  centre_x( a_forme.centre_x ), centre_y( a_forme.centre_y )
{}

Cercle::Cercle( double x = 0, double y = 0, double r = 0 ) :
  Forme2D( x, y ), rayon( r )
{}

Cercle::Cercle( const Cercle & a_cercle ) :
  Forme2D( a_cercle.centre_x, a_cercle.centre_y ), rayon( a_cercle.rayon )
{}
```

Numéro 7. (0 pts)
Objectif(s) :

- **Application des connaissances.**
- **Programmation C++.**

Question : Soit les classes **Forme2D** et **Cercle** déclarées au numéro précédant, et la classe **Forme3D** suivante:

```
class Forme3D : virtual public Forme2D {
 public:
  Forme3D (double x = 0, double y = 0, double z = 0);
  virtual double volume () = 0;
 protected:
  double centre_z;
};
```

Écrivez le code pour les deux classes concrètes (non abstraites) **Cylindre** et **Sphere**. Ces classes doivent contenir un constructeur par défaut. Les classes **Cylindre** et **Sphere** doivent être dérivées des classes **Cercle** et **Forme3D**.

	Cylindre	Sphere
surface	$2\pi rh + 2\pi r^2$	$4\pi r^2$
volume	$\pi r^2 h$	$\frac{4\pi r^3}{3}$

où h est la hauteur du cylindre et $\pi \approx 3.1415926$.
Réponse :

```
class Cylindre : public Cercle, public Forme3D {
 protected :
  double hauteur;
 public :
  Cylindre( double x = 0, double y = 0, double z = 0, double r = 0, double h = 0 ) :
    Forme3D( x, y, z ), rayon( r ), hauteur( h ) {}
  double surface() {
    return 2 * 3.1415926 * rayon * ( hauteur + rayon );
  }
  double volume() {
    return 3.1415926 * rayon * rayon * hauteur;
  }
};
```

```
class Sphere : public Cercle, public Forme3D {
 public :
  Sphere( double x = 0, double y = 0, double z = 0, double r = 0 ) :
    Forme3D( x, y, z ), rayon( r ) {}
  double surface() {
    return 4 * 3.1415926 * rayon * rayon;
  }
  double volume() {
    return ( 4.0 / 3.0 ) * 3.1415926 * rayon * rayon * rayon;
  }
};
```