Trimestre Automne, 2015

Mohamed Lokbani

IFT1169 – Examen Intra –

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

Nom : _____| Prénom(s) : _____|

Signature : _____| Login : _____|

Date : Mercredi 14 octobre 2015

Durée : 2 heures (de 18h30 à 20h30)

Local : Z-310, 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._____/20 (1.1 à 1.10)

2._____/20 (2.1, 2.2, 2.3)

3._____/20 (3.1)

4._____/20 (4.1)

5._____/20 (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 (20 points) répondez par « vrai » ou « faux » en y incluant une très courte explication.

1.1 [VRAI | FAUX] L'édition de liens ne peut pas avoir lieu, si la fonction « `main` » est manquante dans le programme.

1.2 [VRAI | FAUX] La valeur de la variable `a` vaut `20` après l'exécution des deux instructions ci-dessous :

```
int a =37, b = 13;
a = (a>b)?(a+b):(a-b);
```

1.3 [VRAI | FAUX] Soit A une classe de base et B une classe qui dérive de A. Un pointeur (ptr) du type la classe de base (A) ne peut contenir que l'adresse d'un objet de la même classe (ici A).

1.4 [VRAI | FAUX] Une classe virtuelle est la même chose qu'une classe qui contient une fonction virtuelle.

1.5 [VRAI | FAUX] Une fonction amie de la classe C ne peut pas accéder aux membres privés d'une classe qui dérive de C.

1.6 [VRAI | FAUX] Une classe est dite abstraite si toutes ses méthodes sont virtuelles pures.

1.7 [VRAI | FAUX] Si une exception est levée en dehors du bloc « Try », elle provoque l'arrêt du programme.

1.8 [VRAI | FAUX] Il n'est pas possible d'accéder à une variable globale à partir d'un bloc local si nous avons défini dans ce bloc, une variable locale qui porte le même nom que la variable globale.

1.9 [VRAI | FAUX] Il n'est pas possible d'empêcher un programmeur d'accéder au constructeur de recopie et à l'opérateur d'affectation d'une classe donnée?

1.10 [VRAI | FAUX] Inclure un fichier d'en-tête en utilisant <> ou "" c'est la même chose. Exemple :
a- #include <test.h> <=> b- #include "test.h"

Exercice 2 (20 points) le programme suivant compile et s'exécute correctement.

```
 1  #include<iostream>
 2  using namespace std;
 3  class A {
 4  public:
 5         A(){ cout <<"1";}
 6  };
 7  class B: virtual public A {
 8  public:
 9         B(){cout <<"3";}
10  };
11  class C: virtual public A {
12  public:
13         C(){cout<<"5";}
14  };
15  class D:public B,public C {
16  public:
17         D(){cout<<"7";}
18         D(const D & obj){cout <<"8";}
19  };
20  int main() {
21         D d1;
22         D d(d1);
23         cout << endl;
24         return 0;
25  }
```

2.1 Tout en développant votre réponse, que va afficher en sortie le précédent programme?

2.2 On supprime « virtual » de la ligne « 7 » comme suit :

```
class B: public A {
```

Tout en développant votre réponse et en tenant compte de cette modification, que va-t-il afficher en sortie le précédent programme qui compile et s'exécute correctement?

2.3 On retire « virtual » des deux lignes « 7 » et « 11 » pour avoir ce qui suit :

```
class B: public A { …
class C: public A { …
```

Tout en développant votre réponse et en tenant compte des deux modifications, que va-t-il afficher en sortie le précédent programme qui compile et s'exécute correctement?

Exercice 3 (20 points) le programme suivant compile et s'exécute correctement.

```
 1  #include <iostream>
 2  #include <exception>
 3
 4  int zz = 0;
 5  class A {
 6  public:
 7        A() {
 8              std::cout << 'a' << std::endl;
 9              if (zz++ == 0) {
10                    throw std::exception();
11              }
12        }
13        ~A() { std::cout << 'A' << std::endl; }
14  };
15  class B {
16  public:
17        B() { std::cout << 'b' << std::endl; }
18        ~B() { std::cout << 'B' << std::endl; }
19        A a;
20  };
21  void fonction() { static B b; }
22  int main() {
23        try {
24              fonction();
25        }
26        catch (std::exception &) {
27              std::cout << 'c'  << std::endl;
28              fonction();
29        }
30        std::cout << "Fin du main" << std::endl;
31        return 0;
32  }
```

3.1 Tout en développant votre réponse, que va afficher en sortie le précédent programme?

Exercice 4 (20 points) le programme suivant compile correctement.

```
 1  #include <iostream>
 2
 3  using namespace std;
 4
 5  class Exo4 {
 6        enum{ taille=5 };
 7        int *tab;
 8        int valeur;
 9        char *nom;
10   public:
11        Exo4(char *str, int i) : nom(str), valeur(i) {
12              tab = new int[taille];
13              for(int i=0;i<taille;tab[i]=valeur + i++);
14        }
15        ~Exo4() {
16              delete [] tab;
17              cout << "tab de " << nom << " est detruit" << endl;
18        }
19        void print_tab() {
20              cout << nom << ":" << endl;
21              for(int i=0;i<taille;i++){
22                    cout << "Element #" << i;
23                    cout << "=" << get_tab(i) << endl;
24              }
25        }
26        void set_tab(int idx, int val) {
27              tab[idx]=val;
28        }
29        int get_tab(int idx) {
30              return tab[idx];
31        }
32  };
    int main() {
          Exo4 A1("A1",0), B2("B2",10);
          B2=A1;
          A1.set_tab(0, 32);
          B2.set_tab(1, 56);
          A1.print_tab();
          B2.print_tab();

          return 0;
    }
```

4.1 Tout en développant votre réponse, que va afficher en sortie le précédent programme?

Exercice 5 (20 points) soit les 5 fichiers suivants, écrits dans un langage « C++ » :

```
/***********************
* Fichier: Propriete.h *
***********************/
#ifndef Propriete_H
#define Propriete_H

class Propriete {
// etc.
};

#endif
```

```
/*********************
* Fichier: Terrain.h *
*********************/
#ifndef Terrain_H
#define Terrain_H

#include "Propriete.h"

class Terrain : public Propriete {
// etc.
};
#endif
```

```
/*************************
* Fichier: Propriete.cpp *
*************************/

#include "Propriete.h"

// definitions des méthodes
// de la classe Propriete
```

```
/************************
* Fichier: Terrain.cpp *
************************/

#include "Terrain.h"

// definitions des méthodes
// de la classe Terrain
```

```
/*********************
* Fichier: Main.cpp  *
*********************/

#include "Propriete.h"
#include "Terrain.h"

int main() {
//etc.
}
```

5.1 Nous avons reçu le « makefile » ci-dessous. En tentant de l'exécuter, nous avons obtenu une panoplie d'erreurs à l'écran. On vous demande de trouver ces erreurs, de les expliquer puis de les corriger. S'il manque des éléments dans le fichier « makefile », ajoutez-les, tout en expliquant brièvement pourquoi il serait nécessaire de les inclure. À noter que la cible globale du « makefile » est « all ».

```
# Makefile à corriger

Terrain.o: Terrain.cpp Propriete.cpp Terrain.h
gcc -Wall -g -c Terrain.h

Propriete.o: Propriete.cpp Propriete.h
gcc -Wall -g -c Propriete.h

Main.o: Main.cpp Propriete.cpp Terrain.cpp
gcc -Wall -g -c Main.h

all: Main.o Propriete.o Terrain.o
gcc -Wall -g -o test Main.o Propriete.cpp Terrain.cpp
```