Trimestre Hiver, 2011

Mohamed Lokbani

IFT1169 – Examen Intra –

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

Nom : _____| Prénom(s) : _____|

Signature : _____| Code perm : _____|

Date : mardi 22 février 2011

Durée : 2 heures (de 18h30 à 20h30)

Local : Z-200 ; 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 1.11)

2._____/20 (2.1, 2.2)

3._____/20 (3.1 3.2 3.3 3.4)

4._____/30 (4.1 4.2 4.3 4.4)

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 Le sigle **`IFT1169`** correspond au cours :

1.2 Rassembler les différents fichiers objets pour former un exécutable est une opération :

a) de compilation
b) d'assemblage
c) de débogage
d) de linkage

1.3 Si « `Ma_Classe` » a été déjà définie, le code défini « `ici` » :

```
#ifndef Ma_Classe_H

        // ici

#endif »
```

a) est exécuté
b) est compilé
c) n'est pas pris en consideration
d) est débogué

1.4 Que va afficher en sortie, le fragment de code suivant qui compile et s'exécute correctement :

```
char unchar;
char titre[] = "Titanic";
unchar = titre[1];
titre[3] = unchar;
cout << titre << endl;
cout << unchar << endl;
```

1.5 Sachant que le constructeur de la classe « `Essai` » est défini comme suit :

```
Essai(int n){
    a=0;
    x_ptr=new char[n];
}
```

Écrivez le code du destructeur de la classe « `Essai` ».

1.6 À partir du fragment de code suivant, complétez ce qui suit :

```
void Test::rien(int n) {
    int a=2;
    x=n;
    p=new int[n];
    p[0]=2;
}
```

Le nom de la classe est:
Le nom de la méthode est:
La classe contient les membres suivants:

1.7 a) Tout en expliquant brièvement votre réponse, complétez les champs des lignes « 1, 2 et 3 » dans le fragment de code suivant :

```
1   _______ x;

2   _______ y;

3   __________ z = new Rien[4];

4   char s[] = "IFT1169";
5   int i = 65;
6   x = &i;
7   y = *s;
```

b) Pour que le précédent fragment de code puisse compiler correctement, la classe « Rien » doit-elle avoir obligatoirement une méthode particulière? Quelle est cette méthode? Pourquoi? Écrivez le contenu d'une telle méthode.

1.8 Que va afficher en sortie le fragment de code suivant qui compile correctement, et pourquoi?

```
1   int *p1, *p2;
2   p1 = new int;
3   p2 = new int;
4   *p1 = 11;
5   *p2 = 0;
6   p2 = p1;
7   cout << *p1 << " " << *p2 << endl;
```

1.9 Que va afficher en sortie le fragment de code suivant qui compile correctement, et pourquoi?

```
1  int* p1 = NULL ;
2  int* p2 = NULL ;
3  *p1 = 3 ;
4  *p2 = 5 ;
5  p2 = p1 ;
6  cout << *p2 << endl ;
```

1.10 Que va afficher en sortie le fragment de code suivant qui compile et s'exécute correctement, et pourquoi?

```
char a[] = "Ceci est un exercice de l'examen du cours IFT1169.";

char* p = a; // p un pointeur initialisé avec l'adresse « a[0] »

int valeur = 0;
while (*p != '\0') {
        valeur++;
        while(*p != ' ' && *p != '\0') p++;
        while (*p == ' ') p++;
}
cout << "valeur: " << valeur << endl;
```

1.11 Soient les deux instances « obj1 » et « obj2 » d'une classe fictive « Test » qui contient une allocation dynamique de la mémoire pour un de ses attributs. Si cette classe ne définit pas un opérateur d'affection « = », que se passe-t-il si l'instruction suivante est exécutée « obj1=obj2 »?

Exercice 2 (20 points)

2.1 Un programme utilisant la classe « `Essai` » définie ci-dessous, plante à la compilation. Expliquez les raisons de ce plantage en apportant les corrections nécessaires à la classe, sauf la ligne -5- qui ne doit pas être modifiée.

```
1   class Essai{
2   public :
3       Essai(int* e) {x=e;}
4   private:
5       int& x;
6   };
```

2.2 Le fragment de code suivant ne compile pas. Expliquez les erreurs générées. Proposez 3 solutions distinctes pour corriger ces erreurs.

```
1   class X {
2   public:
3       X( const int e) { v = e; }
4   private:
5       int v ;
6   } ;
7   class Y:public X {
8   public:
9       Y() {}
10  } ;
```

Exercice 3 (20 points) Soit le fragment de code suivant :

```
 1  #include <iostream>
 2
 3  using namespace std;
 4
 5  class A {};
 6  class B : public A {};
 7  class C : private A {};
 8
 9  int main() {
10          B* pB = new(B);
11          C* pC = new(C);
12
13          // on ajoute ici une instruction à la fois
14
15          return 0;
16  }
```

3.1 Si l'on ajoute à la **ligne 13** l'instruction suivante :

```
A* pA1 = pB;
```

Le précédent programme va-t-il compiler? Expliquez votre réponse.

3.2 Si l'on ajoute à la **ligne 13** l'instruction suivante :

```
A* pA2 = dynamic_cast<A*>(pC);
```

Le précédent programme va-t-il compiler? Expliquez votre réponse.

3.3 Si l'on ajoute à la **ligne 13** l'instruction suivante :

```
A* pA3 = (A*) pC;
```

Le précédent programme va-t-il compiler? Expliquez votre réponse.

3.4 Si l'on ajoute à la **ligne 13** l'instruction suivante :

```
A* pA4 = reinterpret_cast<A*>(pC);
```

Le précédent programme va-t-il compiler? Expliquez votre réponse.

Exercice 4 (30 points) Une image numérique est représentée par un ensemble de pixels. Chaque pixel est un petit rectangle à qui l'on associe une couleur. Cette couleur est représentée par les trois composantes : « rouge », « bleue » et « verte ». Chacune de ces trois composantes peut avoir une valeur entière comprise entre 0 et 255.

Ainsi donc, un pixel blanc est représenté par les valeurs (255, 255, 255). Dans ce cas, la valeur du rouge est égale à 255, ainsi que celles du bleu et du vert. Un pixel noir est représenté par les valeurs (0, 0, 0). Dans ce cas, la valeur du rouge est égale à 0, ainsi que celles du bleu et du vert.

Le but de cet exercice est de créer une classe simplifiée « Pixel » et de lui associer une série de méthodes au besoin. N'utilisez « friend » que si c'est nécessaire.

4.1 Définissez la classe Pixel avec les attributs tels que décrits précédemment et les différentes méthodes **nécessaires** pour que les deux instructions ci-dessous puissent s'exécuter correctement :

```
1 | Pixel p1(255,255,255); // Pixel blanc
2 | Pixel p2; // Pixel par défaut noir (0, 0, 0)
```

Vous devez écrire le code des méthodes pour le bon fonctionnement des instructions 1 et 2, à l'extérieur de la classe.

4.2 a) Ajoutez dans la classe Pixel, la déclaration de la méthode permettant cette instruction :

```
1 | cout << p1; // p1 est 255, 255, 255
```

Mentionnez un numéro de ligne « fictif » dans la classe Pixel où il faudra insérer cette déclaration.

b) Écrire le code complet d'une telle méthode afin d'obtenir l'affichage suivant, pour l'exemple précédent :

```
Le Pixel est représenté par les 3 couleurs suivantes:
Rouge : 255
Bleue : 255
Verte : 255
```

4.3 Nous voulons additionner, en moyennant deux pixels P1 et P2, comme suit :

```
Pixel P1           : rouge = 100, vert = 200, bleu = 50
Pixel P2           : rouge = 200, vert = 150, bleu = 90
Pixel P3 = P1 + P2 : rouge = 150, vert = 175, bleu = 70
```

a) Ajoutez dans la classe Pixel, la déclaration de la méthode permettant une telle opération. Mentionnez un numéro de ligne « fictif » dans la classe Pixel où il faudra insérer cette déclaration.

b) Écrivez le code complet d'une telle méthode.

4.4 Nous voulons additionner un pixel avec un entier positif, comme suit :

```
Pixel P1 : rouge = 100, vert = 200, bleu = 50
x        : 10
P1 + x   : rouge = 110, vert = 210, bleu = 60
```

On s'assure que la valeur de chaque couleur ne déborde pas de la limite autorisée (255). Si c'est le cas, on ne garde que le reste de la division par 255 : 100+600=700/255, couleur=~~200~~ 190

a) Ajoutez dans la classe Pixel, la déclaration de la méthode permettant une telle opération. Mentionnez un numéro de ligne « fictif » dans la classe Pixel où il faudra insérer cette déclaration.

b) Écrivez le code complet d'une telle méthode.