Trimestre Automne, 2005

Mohamed Lokbani

IFT1166 – Examen Final

Inscrivez tout de suite votre nom et code permanent.

Nom: _____| Prénom(s): _____|

Signature: _____| Code perm: _____|

Date : 10 décembre 2005

Durée: 2 heures et 45 minutes (de 09h30 à 12h15) Local: Z-317, du Pavillon Claire-McNicoll.

Directives:

- Toute documentation permise.
- Calculatrice **non** permise.

- Répondre directement sur le questionnaire.
- Les réponses **doivent être brèves, précises et clairement présentées.**

1._____/15 (1.1, 1.2, 1.3, 1.4)

2._____/20 (2.1)

3._____/20 (3.1, 3.2, 3.3, 3.4)

4._____/15 (4.1, 4.2, 4.3, 4.4)

5_____ /30 (5.1, 5.2, 5.3)

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 remet sa carte d'étudiant au surveillant, l'absence ne devant pas dépasser 5 minutes.

* Un seul étudiant à la fois peut quitter la salle.

* Tout plagiat, copiage ou fraude constitue une infraction et l'étudiant qui commet ce délit se voit attribuer la note F par le professeur.

F.A.S

Exercice 1 (15 points)

1.1 Un des gros problèmes dans le développement de logiciels est la réutilisation d'un travail déjà réalisé. Expliquer brièvement comment la programmation par objets peut aider à solutionner ce problème de réutilisation.

1.2 Soit la classe de base (B) « `class B{int a;};` ». Doit-on ajouter quelque chose de particulier au niveau de cette classe de base pour que son membre privé « `a` » ne soit accessible qu'à partir d'une classe dérivée (D)? Si oui, quoi au juste? (Expliquer votre réponse)

1.3. Que signifient les termes "une variable de classe" et "une méthode de classe"?

1.4. Tout en justifiant votre réponse, quelle est la valeur de l'expression suivante pour a différent de b (a!=b):

```
((a<b)?a:b)+((a>b)?b:a))/2
```

A) La plus petite valeur de a et b?

B) La plus grande valeur de a et b?

C) La moyenne de a et b?

D) La somme de a et b?

La bonne réponse est _____ parce que ……

Exercice 2 (20 points) Que va afficher en sortie le programme suivant qui compile et s'exécute correctement :

```
 1      #include <iostream>
 2      using namespace std;
 3
 4      class A {
 5        public:
 6          A(){ cout << "CD-A-\n"; }
 7          A(A &un_a){ cout << "CR-A-\n"; }
 8          ~A() { cout << "D-A-\n"; }
 9      };
10      class B : A {
11          int bval;
12        public:
13          B() : bval(0){ cout << "CD-B-\n"; }
14          B(int i):bval(i) { cout << "Cint-B-\n"; }
15          B(B &un_b):bval(un_b.bval) { cout << "CR-B-\n"; }
16          ~B() { cout << "D-B- " << bval << endl; }
17      };
18      void Test(A avar, B &bvar){
19          cout << "Dans Test\n";
20      }
21      int main(){
22          A avar;
23          B bvar(10);
24          cout << "Dans main\n";
25          Test(avar, bvar);
26          cout << "Quittes main\n";
27          return 0;
28      }
29
```

Attention : vous devez expliquer et détailler votre réponse.

Exercice 2 (suite de votre réponse)

Exercice 3 (20 points) Pour le fragment de code suivant, indiquer si les lignes 4 dernières lignes provoquent une erreur à la compilation. S'il y a erreur, vous devez la commenter. Dans le cas contraire, tout en expliquant votre réponse, vous devez donner l'affichage en sortie. À noter, que si une des lignes provoque une erreur de compilation, cette dernière ne va pas planter le reste du programme. Après avoir expliqué l'origine de l'erreur, vous pouvez la considérer comme en commentaire pour le reste du programme. Ceci va vous permettre de vérifier les 4 lignes de manière indépendante.

```
#include <iostream>

using namespace std;

class B{};
class C{};
class BBB:public B{};
class CCC:public C{};

B &b1ref = * (new B);
B &b3ref = * (new BBB);
BBB bbb;
C &c1ref = * (new C);
C &c3ref = * (new CCC);
CCC ccc;

void  fn(B,C){cout << "fn1" << endl;}
void  fn(B,CCC){cout << "fn2" << endl;}
void  fn(BBB,C){cout << "fn3" << endl;}

void foo()
{
      fn(b1ref,c1ref); // Ligne -1-
      fn(b1ref,c3ref); // Ligne -2-
      fn(b3ref,c3ref); // Ligne -3-
      fn(bbb,ccc);     // Ligne -4-
}
```

3.1 `fn(b1ref,c1ref); // Ligne -1-`

☐ correcte ☐ incorrecte

Pourquoi (affichage en sortie)

3.2 `fn(b1ref,c3ref); // Ligne -2-`

☐ correcte ☐ incorrecte
Pourquoi (affichage en sortie)

3.3 `fn(b3ref,c3ref); // Ligne -3-`

☐ correcte ☐ incorrecte
Pourquoi (affichage en sortie)

3.4 `fn(bbb,ccc); // Ligne -4-`

☐ correcte ☐ incorrecte
Pourquoi (affichage en sortie)

Exercice 4 (15 points) Soit la méthode suivante :

```
double caFaitQuoi(double a,int b){
    if (b==0)
        return 1;
    else
        return(a*caFaitQuoi(a,b-1));
}
```

4.1 Cette méthode est elle récursive ? (Expliquer brièvement pourquoi)

4.2 Indiquer la valeur de « n » après l'exécution de chacune des instructions suivantes (on suppose que « n » est déclaré entier) :

Appel	Résultat
`n = caFaitQuoi(0,2);`	
`n = caFaitQuoi(2,4);`	

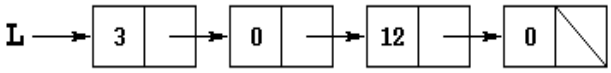
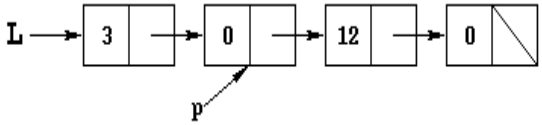
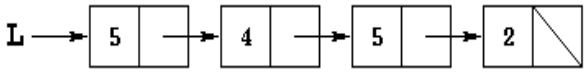
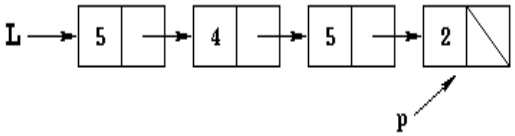
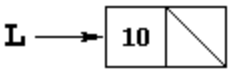
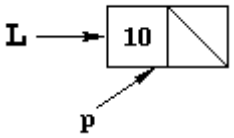
4.3 Énoncer en une phrase **simple** ce que calcule cette méthode.

4.4 La méthode « caFaitQuoi » butte sur un bug quand la valeur de « b » est négative. Expliquer pourquoi ? Modifier par la suite la méthode « caFaitQuoi » pour qu'elle produise un résultat correct quelle que soit la valeur de « b » (b positif, nul ou négatif).

Exercice 5 (30 points) Nous allons supposer l'implémentation suivante d'une liste chaînée de nombres entiers :

```
struct ListeNoeud{
    int         info;
    ListeNoeud * suiv;
    ListeNoeud(int val,ListeNoeud *ptr):info(val),suiv(ptr){ }
};
```

5.1 Si « x » est la plus petite valeur dans la liste ; la méthode « PremierMin », dont l'en-tête est donné ci-dessous, retourne un pointeur vers la première occurrence de « x ». Si la liste est vide « PremierMin » retourne une valeur nulle.
Par exemple, si « L » et « p » sont des variables « ListeNoeud* » :

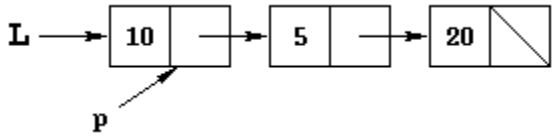
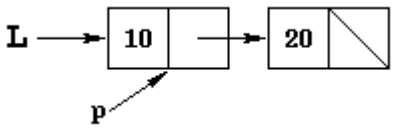
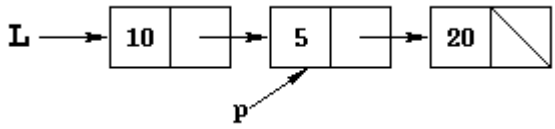
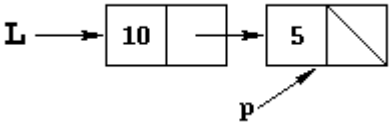
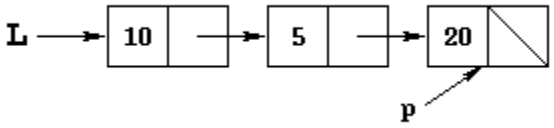
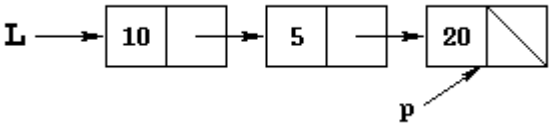
Liste chaînée *L*	**Résultat suite à « *p = PremierMin(L)* »**
	
	
	

```
ListeNoeud* PremierMin(ListeNoeud *liste){




}
```

5.2 Écrire la méthode « RetirerSuivant », dont l'en-tête est donné ci-dessous. Cette méthode accepte l'argument « p », un pointeur non nul du type « ListeNoeud ». La méthode « RetirerSuivant » retire de la liste le nœud qui suit le nœud pointé par « p » et libère en conséquence l'espace mémoire alloué. La méthode « RetirerSuivant » n'a aucun effet si le nœud pointé par « p » est le dernier nœud de la liste.

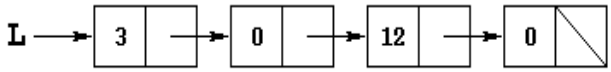
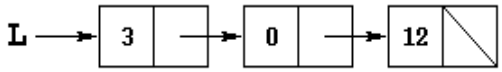
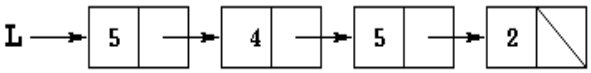
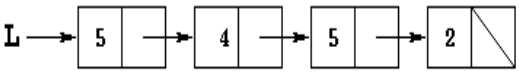
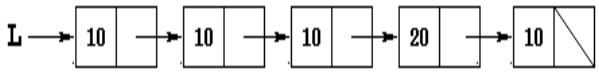
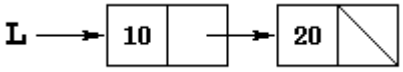
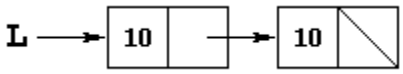
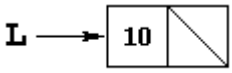
Avant l'appel à *RetirerSuivant(p)*	**Après l'appel à *RetirerSuivant(p)***
	
	
	

```
void RetirerSuivant(ListeNoeud * ptr)
// précondition: ptr != NULL
{




}
```

5.3 Si « x » est la plus petite valeur dans la liste ; la méthode « RetirerDupMins » dont l'en-tête est donné ci-dessous, élimine toutes les occurrences de « x » sauf la première occurrence de « x ». La méthode n'a aucun effet si la liste est vide.

Liste *L* avant l'appel à *RetirerDupMins(L)*	**Liste *L* après l'appel à *RetirerDupMins(L)***
	
	
	
	

Dans l'écriture de "RetirerDupMins" vous pouvez faire appel aux deux méthodes « PremierMin » et « RetirerSuivant » telles que décrites dans les questions (5.1 et 5.2), même si vous ne les avez pas codées encore.

```
void RetireDupMins(ListeNoeud * liste){




}
```

Joyeuses Fêtes !