IFT1166 - Session Hiver 2000, Final

Mohamed Lokbani

Nom:_____| Prénom(s):_____|

Signature:_____| Code perm:_____|

Date:19 avril 2000

Durée: 3 heures (de 18h30 à 21h30) Local: B-2285

Directives:

- Il vous est permis d'utiliser un livre de votre choix.
- Les documents de cours ne sont **pas** autorisés (**y compris mes e-notes disponibles sur la page web du cours ou à la bibliothèque Math/Info**).
- Ordinateurs personnels prohibés.
- Calculatrice (simple) permise.

- Répondre directement sur le questionnaire.

Bon courage!

1._____/15

2._____/10

3._____/20

4._____/15

5._____/20

6._____/20

Total:_____/100

Bonus._____/10

Question 1 (15 points)

La fonction `remplir(…)`, lit des entiers d'un fichier et les range dans un tableau d'entiers. La fonction `remplir()` a 3 arguments:

1. un stream ouvert en lecture,
2. un tableau d'entiers,
3. un entier indiquant le nombre d'éléments lus.

Supposez que le code suivant a été exécuté avec succès:

```
const int max = 100; // le nombre maximal d'éléments
int item[max];       // le tableau des éléments
int nb_item = 0;     // le nombre d'éléments
char NomFic[80];     // le nom de fichier
cin >> NomFic;       // lecture du nom de fichier
ifstream in(NomFic); // flux en entrée associé au fichier NomFic
```

Lequel des appels suivants est correct? et pourquoi? (vous pouvez choisir une ou plusieurs réponses).

Q1.A- `remplir(in, item[], nb_item);`

vrai ☐ faux ☐

pourquoi?

Q1.B- `remplir(NomFic, item[], nb_item);`

vrai ☐ faux ☐

pourquoi?

Q1.C- `remplir(in, item, nb_item);`

vrai ☐ faux ☐

pourquoi?

Q1.D- `remplir(ifstream, item, nb_item);`

vrai ☐ faux ☐

pourquoi?

Q1.E- `remplir(NomFic, item, max);`

vrai ☐ faux ☐

pourquoi?

Question 2 (10 points)

Étant donné la fonction:

```
float RacineCarree(float w)
{
// calcule la racine carrée avec sqrt, et retourne le résultat.
}
```

Écrire le prototype de la fonction générique (Template) `RacineCarree` permettant de traiter les combinaisons suivantes:

Type en Argument	Type de retour
int	int
int	float
float	int
float	float
float	double
double	float

Question 3 (20 points)

Soit le programme suivant:

```
#include <iostream.h>

class X {
public:
  X() {a = 0;}
  X(int valeur):a(valeur) {}
  X(const X& zz) {a = zz.a;}
  X& operator=(const X& zz) {
    if (this!=&zz) {
      a = zz.a;
    }
    return *this;
  }
  void Print() const {
    cout << a << endl;
  }
private:
  int a;
};


int main() {

  X v,w(1985);          // ligne: 1

  v.Print();            // ligne: 2
  w.Print();            // ligne: 3

  X *pa = new X(v);     // ligne: 4
  pa->Print();          // ligne: 5

  X *pb = new X(1949);  // ligne: 6
  pb->Print();          // ligne: 7

  v = w;                // ligne: 8
  v.Print();            // ligne: 9

  pa = pb;              // ligne: 10
  pa->Print();          // ligne: 11

  *pb = 2010;           // ligne: 12
  pb->Print();          // ligne: 13

  return 0;             // ligne: 14

}
```

Q3.A- Donner les lignes de la fonction main où sont appelées les fonctions membres de la classe `X`, en indiquant à chaque fois le nom de l'objet crée (ou receveur). (Si plusieurs objets sont créés pour une même ligne, indiquez-les tous).

	ligne 1	**ligne 4**	**ligne 6**	**ligne 8**	**ligne 10**	**ligne 12**
X::X()						
X::X(int)						
X::X(const X&)						
X::operator=(const X&)						

Q3.B- Donner les résultats obtenus en sortie après l'appel des lignes suivantes, de la fonction main:

	ligne 2	**ligne 3**	**ligne 5**	**ligne 7**	**ligne 9**	**ligne 11**	**ligne 13**
sortie							

Soit **H** le nom de l'objet crée à la ligne 4. Sur cette même ligne (4), **H** est accessible par le pointeur **pa**. Après avoir exécuté la ligne 10, et avant d'avoir exécuté la ligne 11:

Q3.C.1- `H` existe toujours mais n'est plus accessible?

vrai ☐ faux ☐

pourquoi?

Q3.C.2- `H` existe et `H.a` a pris la valeur 2010?

vrai ☐ faux ☐

pourquoi?

Question 4 (15 points)

Soit la classe générique (Template) suivante:

```
template<class Type1, class Type2>
class ClasseBase
{
protected:
      Type1 Champ1;
      Type2 Champ2;
public:
      ClasseBase() {}
      ClasseBase(Type1 arg1, Type2 arg2) :
            Champ1(arg1), Champ2(arg2) {}
      virtual void Print()
            { cout << Champ1 << ' ' << Champ2 << endl; }
};
```

Q5.A- En fonction de la précédente définition de la classe template `ClasseBase<Type1,Type2>`, citez au moins un type qu'elle ne pourra pas traiter correctement (justifiez votre réponse).

Q5.B- Écrire la définition d'une classe dérivée `ClasseDerivee`, qui dérive publiquement de la classe de base `ClasseBase` et qui contient les membres suivants:

- un membre donnée `Champ3` du type `Type2`
- un seul constructeur
- une fonction `Print()` pour afficher les membres données de la classe de base et de la classe dérivée.

Q5.C- Complétez la fonction main suivante:

```
int main()
{
      int x = 40;
      float y = 55.55;
      char a = 'w', b='z';

      // À titre d'exemple
      // création (instantiation) de l'objet BaseExpObj du type
      // ClasseBase<int,int> et cela sans arguments.

      ClasseBase<int,int> BaseExpObj();


      // À ÉCRIRE
      // création (instantiation) de l'objet ObjBase du type ClasseBase
      // avec les valeurs x et y comme arguments.




      // À ÉCRIRE
      // création (instantiation) de l'objet ObjDerivee du type ClasseDerivee,
      // avec les valeurs a,b comme arguments.




      // À ÉCRIRE
      // appel de la fonction Print de la classe de base sur l'objet ObjBase.




      // À ÉCRIRE
      // appel de la fonction Print de la classe dérivée sur l'objet ObjDerivee



      return 0;
}
```

Question 5 (20 points)

Soit le programme suivant:

```
#include <iostream.h>

class X {
public:
  int a;

  X(int valeur):a(valeur) {}
  void f0() {
    cout << "X::f0" << endl;
  }
  virtual void f1() {
    cout << "X::f1" << endl;
  }
};

class Y:public X {
public:
  Y(int zz):X(zz) {}
  void f1() {
    cout << "Y::f1" << endl;
    f0();
  }
};

int main() {

  X x(0);
  Y y(1);

  X *px;
  Y *py;

  py = &y;
  y.f1();               // ligne 1
  py->f1();             // ligne 2

  px = &y;
  y.f1();               // ligne 3
  px->f1();             // ligne 4

  // px = py;           // ligne 5
  // py = px;           // ligne 6

  px = &y;px->f1();     // ligne 7

  px = &x;px->f1();     // ligne 8

  py = &y;py->f1();     // ligne 9

  return 0;
}
```

Q4.A- Les lignes 1 et 2 écrivent-elles la même chose?

oui ☐ non ☐

qu'est ce qu'elles affichent?

Q4.B- Les lignes 3 et 4 écrivent-elles la même chose?

oui ☐ non ☐

qu'est ce qu'elles affichent?

Q4.C- Indiquer toutes les fonctions qui seront appelées aux lignes suivantes:

	X::f0()	**X::f1()**
ligne 7		
ligne 8		
ligne 9		

Q4.D- Que se passe-t-il si on enlève les commentaires des lignes 5 et 6? Les transformations polymorphiques (conversions de types) sont-elles correctes?

Q4.D.1- `px = py; // ligne 5`

oui ☐ non ☐

pourquoi?

Q4.D.2- `py = px; // ligne 6`

oui ☐ non ☐

pourquoi?

Question 6 (20 points)

Cet exercice à pour but de vous faire programmer une méthode de tri. L'algorithme que vous devez programmer triera les éléments d'un tableau dans un ordre décroissant. Pour chaque position du tableau, vous devez exécuter les deux opérations suivantes:

1- la recherche dans le tableau de l'élément le plus grand (e_{max}),
2- la permutation de e_{max} avec l'élément pointé par la position courante.

Soit le tableau `tab` des entiers contenant ce qui suit: `int tab[5] = {7,6,8,2,5};`

La table suivante illustre le fonctionnement de ce tri pour le tableau `tab`

tab	**position**	**e_{max}**
7 6 8 2 5	0	8
8 6 7 2 5	1	7
8 7 6 2 5	2	6
8 7 6 2 5	3	5
8 7 6 5 2	4	2

Dans la colonne `tab`, les éléments en caractères gras, représentent les éléments déjà triée.

résultat du tri: **{8,7,6,5,2}**

Q6.A

Écrire une fonction permettant de trier un tableau **d'entiers** `tab` de dimension `taille`, en utilisant pour cela ces consignes de programmation:

-1- vous devez utiliser l'algorithme de tri précédemment expliqué.
-2- vous ne devez pas utiliser d'autre tableau que le tableau original pour réaliser le tri.
-3- vous pouvez si vous le souhaiter découper votre code en plusieurs fonctions.
-4- vous ne devez pas utiliser les STL.

Q6.B Modifier votre fonction afin de permettre de trier aussi bien le type **`int`**, le type **`char`** `et` le type **`string`**.

Q6.C Modifier votre fonction de tri afin de permettre de trier le type **`char*`**. À titre d'exemple d'un tableau **`char*`**:

```
char* tab[5]={"safran","voile","foc","gouvernail","catamaran"};
```

Bonus
(10 points)

Pour obtenir le bonus, il faut que vous répondiez (correctement!) **à toutes les questions** de l'exercice de votre choix. Donc du 0 ou 10.

Question Bonus (10 points)

Soit le programme suivant:

```
#include <iostream.h>
#include <algorithm>
#include <functional>

using namespace std;

class f_pair:public unary_function <unsigned int,bool> {
public:
  bool operator() (unsigned int n) const {return n%2 == 0;}
};

int main() {

  int tab[] = {1,2,3,4,5,6,7};

  int n = count_if(tab,tab+5,f_pair());   // ligne 1
  cout << n << endl;                      // ligne 2
  return 0;
}
```

QB.1- Expliquez le but du programme précédent.

QB.2- Donner le contenu de **n** affiché en sortie après exécution de la ligne 2. (justifier votre réponse)

QB.3- En utilisant un appel à la fonction **`count_if`** de la même manière qu'à la ligne 1, écrivez l'instruction qui permet de réaliser l'inverse de ce que fait la ligne 1 (inverse du critère):
n = count_if(tab,tab+5,le_critère_à_compléter).

QB.4 Et que va valoir **n** dans ce cas?