Trimestre Hiver, 2013

Mohamed Lokbani

IFT1169 – Examen Final –

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

Nom : _____| Prénom(s) : _____|

Signature : _____| Code perm : _____|

Date : mardi 30 avril 2013

Durée : 3 heures (de 18h30 à 21h30)

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._____/10 (1.1, 1.2, 1.3, 1.4)

2._____/20 (2.1, 2.2)

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

4._____/20 (4.1, 4.2, 4.3, 4.4)

5._____/10 (5.1, 5.2)

6._____/20 (6.1)

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 (10 points)

1.1 [VRAI | FAUX] Les « STL » est un des thèmes enseignés dans le cours IFT1169.

1.2 Quelle est la différence entre « wxFrame » et « wxDialog »?

1.3 La méthode « Destroy » est utilisée pour détruire une boite de dialogue. Que va-t-il arriver à toutes les allocations mémoires effectuées dans votre programme suite à l'appel de « Destroy »? Une fuite de mémoire? « Destroy » va faire le ménage proprement? Développez votre réponse.

1.4 Dessinez la boite de dialogue associée à ce fragment de code :

```
1  CustomDialog::CustomDialog(const wxString & title)
2         : wxDialog(NULL, -1, title, wxDefaultPosition, wxSize(250, 230)){
3
4    wxPanel *panel = new wxPanel(this, -1);
5    wxBoxSizer *vbox = new wxBoxSizer(wxVERTICAL);
6    wxBoxSizer *hbox = new wxBoxSizer(wxHORIZONTAL);
7    wxStaticBox *st = new wxStaticBox(panel, -1, wxT("Couleurs"), \
8        wxPoint(5, 5), wxSize(240, 150));
9    wxRadioButton *rb = new wxRadioButton(panel, -1, \
10       wxT("256 Couleurs"), wxPoint(15, 30), wxDefaultSize, wxRB_GROUP);
11   wxRadioButton *rb1 = new wxRadioButton(panel, -1, \
12       wxT("16 Couleurs"), wxPoint(15, 55));
13   wxRadioButton *rb2 = new wxRadioButton(panel, -1, \
14       wxT("2 Couleurs"), wxPoint(15, 80));
15   wxRadioButton *rb3 = new wxRadioButton(panel, -1, \
16       wxT("Personalisé"), wxPoint(15, 105));
17   wxTextCtrl *tc = new wxTextCtrl(panel, -1, wxT(""), wxPoint(95, 105));
18   wxButton *okBouton = new wxButton(this, -1, wxT("Ok"),
19       wxDefaultPosition, wxSize(70, 30));
20   wxButton *fermerBouton = new wxButton(this, -1, wxT("Fermer"),
21       wxDefaultPosition, wxSize(70, 30));
22   hbox->Add(okBouton, 1);
23   hbox->Add(fermerBouton, 1, wxLEFT, 5);
24   vbox->Add(panel, 1);
25   vbox->Add(hbox, 0, wxALIGN_CENTER | wxTOP | wxBOTTOM, 10);
26   SetSizer(vbox);
27   Centre();
28   ShowModal();
29   Destroy();
30 }
```

Expliquez brièvement comment vous êtes arrivés à cette conclusion.

Exercice 2 (20 points)

2.1 Il y a 4 types de relations entre les classes et leurs amis quand nous avons affaire à des templates :

- Un-Plusieurs: une fonction non-template peut-être amie à toutes les instances de la classe template.
- Plusieurs-Un: toutes les instances d'une fonction template peuvent-être amies à une classe régulière non-template.
- Un-Un: une fonction template instanciée avec un jeu d'arguments templates peut-être amie à une classe template instanciée avec le même jeu d'arguments templates.
- Plusieurs-Plusieurs: toutes les instances d'une fonction template peuvent-être amies à toutes les instances d'une classe template.

```
 1  class X{
 2     template<class W> friend int k();
 3  }
 4
 5  template<class A> i();
 6
 7  template<class T> class Y {
 8     friend int g();
 9     friend int h(T);
10     friend int i<T>();
11     template<class V> friend int j();
12  };
```

Ainsi donc, par exemple, la fonction « g() » a une relation « un-plusieurs » avec la classe « Y ». La fonction « g », est non-template et, est amie à toutes les instances de la classe « Y ». Quelles sont la nature des relations des autres fonctions?

La fonction h :

La fonction i :

La fonction j :

La fonction k :

2.2 Citez trois avantages à utiliser des templates.

Exercice 3 (20 points)

La classe Pièce représente une pièce de monnaie canadienne. Elle contient deux membres : cent et dollar. Cette classe contient un ensemble de constructeurs et d'opérateurs surchargés pour permettre l'exécution de cette fonction « main » :

```
 1  int main() {
 2          Piece p1(2, 98), p2(15, 2), p3;
 3
 4          p3 = p1 + p2;
 5          cout << p1 << " + " << p2 << " = " << p3 << endl;
 6          cout << (p1 != p2) << endl;
 7          cout << (p1 > p2) << endl;
 8          cout << (p1 < p2) << endl;
 9          cout << (p1 <= p2) << endl;
10          cout << -p2 << endl;
11
12          return 0;
13  }
```

3.1 La ligne « 2 » fait appel à un ensemble de constructeurs, citez-les en mentionnant leur signature (la déclaration uniquement).

3.2 Les lignes « 4 » à « 10 » font appel à une série d'opérateurs, citez-les en mentionnant leur signature.

Ligne 4 :

Ligne 5 :

Ligne 6 :

Ligne 7 :

Ligne 8 :

Ligne 9 :

Ligne 10 :

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

Exercice 4 (20 points)

4.1 Expliquez à quoi sert le programme suivant qui compile et s'exécute correctement :

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  int main() {
7          vector <int> intvec;
8          int i;
9          while (cin >> i) intvec.push_back(i);
10         for (unsigned int k = 0; k < intvec.size(); i++)
11                 cout << intvec[k] << endl;
12         return 0;
13 }
```

4.2 Expliquez à quoi sert le programme suivant qui compile et s'exécute correctement :

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  int main() {
7          vector <int> intvec;
8          int i;
9          while (cin >> i) intvec.insert(intvec.begin(), i);
10         for (unsigned int k = 0; k < intvec.size(); k++)
11                 cout << intvec[k] << endl;
12         return 0;
13 }
```

4.3 Expliquez à quoi sert le programme suivant qui compile et s'exécute correctement :

```
1  #include <iostream>
2  #include <set>
3  using namespace std;
4
5  int main() {
6          set <int> intvec;
7          set <int>::iterator it;
8          int i;
9          while (cin >> i) intvec.insert(i);
10         for (it = intvec.begin(); it != intvec.end(); it++)
11                 cout << *it << endl;
12         return 0;
13 }
```

4.4 Pensez-vous que le choix de la « STL » dans les trois précédents programmes était judicieux? Développez votre réponse.

Exercice 5 (10 points)

5.1 La fonction suivante est utilisée uniquement pour calculer la moyenne pondérée d'un vecteur donné.

```
1  double unefonction(vector <double> v) {
2          unsigned int i;
3          double total;
4          if (v.size() == 0) return 0;
5          for (i = 0; i < v.size(); i++) v[i] *= ((double) (i+1));
6          total = 0;
7          for (i = 0; i < v.size(); i++) total += v[i];
8          return total / (double) v.size();
9  }
```

La fonction est plus efficace si on lui avait passé la variable « v » par référence, expliquez pourquoi?

5.2 Tout en gardant en tête l'utilité finale de la précédente fonction, récrivez-la en tenant compte maintenant du fait que la variable « v » est fournie par référence.

```
double unefonction(vector <double>& v) {
```

Exercice 6 (20 points)

Un programmeur a développé un programme qui affiche ce qui suit :

```
Fleur *** Rose
Fruit *** Banane
Fleur *** Tulipe
Fleur *** Jasmin
Legume *** Salade
Legume *** Pomme-de-terre
Fruit *** Pomme
Fleur *** Lys
Legume *** Pomme-de-terre
Fruit *** Raisin-rouge
Fruit *** Raisin-sec
```

Vous constatez que l'affichage est un peu tordu! Certains éléments sont doublons, l'affichage est désordonné, etc. Il vous est demandé de relire les données afin de les afficher sous la forme suivante :

```
Fleur: Jasmin Lys Rose Tulipe
Fruit: Banane Pomme Raisin-rouge Raisin-sec
Legume: Pomme-de-terre Salade
```

On regroupe ainsi tous les éléments communs. On élimine les doublons. Finalement, on trie les valeurs associées dans un ordre lexicographique.

Le programme sera exécuté comme suit :

```
Exo6.exe < data_entree.txt
```

L'affichage se fait sur la console.

Choisissiez la bonne représentation STL pour vos données et, assurez-vous de faire une manipulation optimisée de vos données et éviter ainsi de faire des copies à tort et à travers. Il sera tenu compte lors de la correction.

Bon été!