Trimestre Hiver, 2008

Mohamed Lokbani

IFT1169 – Examen Final –

Inscrivez tout de suite votre nom et le code permanent.

Nom: _____| Prénom(s): _____|

Signature: _____| Code perm: _____|

Date : mardi 22 avril 2008

Durée : 3 heures (de 16h30 à 19h30)

Local : Z-209 ; Pavillon Claire McNicoll

Directives:

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

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

1._____/18 (1.1, 1.2, 1.3, 1.4, 1.5, 1.6)

2._____/15 (2.1)

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

4._____/20 (4.1, 4.2, 4.3, 4.4, 4.5)

5._____/13 (5.1)

6._____/14 (6.1, 6.2, 6.3)

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 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 (18 points)

1.1 En C++, nous utilisons la directive « `#ifndef XYZ_H` » au début d'un fichier d'en-tête :

a) afin d'inclure le fichier « .cpp » seulement si « `XYZ_H` » est définie.
b) pour s'assurer que le contenu du fichier d'en-tête ne soit inclus qu'une seule fois.
c) parce que le compilateur « gcc » l'exige.
d) pour rendre la lecture du programme (code) plus lisible.

Laquelle des 4 descriptions est correcte ?

1.2 Quelle est la différence entre les directives « **`#include <xyz>`** » et « **`#include "xyz"`** » ?

1.3 Expliquez la phrase suivante : « Les conteneurs séquentiels se démarquent des tableaux dans la mesure où ils ont la capacité de s'étendre et de diminuer au gré, respectivement des ajouts et des suppressions d'éléments. »

1.4 Vous voulez réaliser une application de suivi de rendez-vous (calendrier électronique) pour votre agenda électronique. Cette application va vérifier en permanence l'état de vos rendez-vous en vous alertant s'il y a lieu que vous avez éventuellement des choses à réaliser. Parmi les représentations suivantes, laquelle est la plus appropriée pour représenter les données de votre application de manière efficace : « `Liste, Table de hachage, Pile, Queue, Queue Prioritaire` » et dites-le pourquoi.

1.5 Vous voulez construire une application de fouille pour collecter des adresses de courriel. Cette application va scanner votre disque dur à la recherche de toutes les adresses de courriel se trouvant sur votre disque. Par la suite, elle va se charger d'envoyer les courriels recueillis à une tierce personne. Parmi les conteneurs suivants « `list, hash_set, vector, map` », lequel est le plus approprié pour contenir les données collectées et dites-le pourquoi ?

1.6 Si « `s` » est du type « `string` », pourquoi faut-il faire attention quand vous faites ce genre de lecture « `cin >> s` » ? Justifiez brièvement votre réponse.

Exercice 2 (15 points) Dans le chapitre « 5, la surcharge des opérateurs », nous avons mentionné dans le paragraphe « 10.1, page 107 » à propos de la surcharge de l'opérateur d'affectation « = », que ce dernier doit être une fonction membre de la classe. Expliquez la motivation en vous inspirant de l'exemple ci-dessous.

```
01  class Test {
02  public:
03      char *p;
04  };
05
06  void fonctUn(Test& arg ) {
07    Test rien;
08    arg = rien;
09  }
10
11  void operator=( Test& dest, const Test& src ) {
12    // Le code complet permettant d'affecter src à dst
13  }
14
15  void fonctDeux(Test& arg){
16    Test rien;
17    arg = rien;
18  }
19
20  int main() {
21    Test xyz;
22
23    fonctUn( xyz );
24    fonctDeux( xyz );
25    // blablabla d'autres choses …
    }
```

Exercice 3 (20 points) Supposez que dans la fonction « main » de votre programme « C++ » deux threads ont été déclarés, et ces deux threads font appel à la même fonction « UnTraitement ». Soit le fragment de code suivant :

```
int balance = 0;
void *UnTraitement(void *arg) {
    char *sbNom;
    sbNom = (char *)arg;
    FaireUnDepot(10);
    cout   <<   "Balance   =   "   <<   balance   <<   \
               " dans le thread " << sbNom << "\n";
    return NULL;
}

int FaireUnDepot(int montant) {
    int ancbalance = balance;
    balance = ancbalance + montant;
    return ancbalance;
}
```

La fonction « UnTraitement » est la fonction exécutée par un thread donné. Elle sert à afficher la balance du thread en cours.

La fonction « FaireUnDepot » sert à déposer un certain « montant » dans un compte, à calculer la nouvelle balance et retourner l'ancienne valeur vers la fonction appelante.

3.1 Expliquez le comportement des deux threads pour le scénario suivant :

Thread 1	Thread 2	Balance
`UnTraitement("A")`		0
`FaireUnDepot(10)`		0
`ancbalance = balance // 0`		0
`balance = ancbalance + montant`		10
`return ancbalance // 0`		10
	`UnTraitement("B")`	10
	`FaireUnDepot(10)`	10
	`ancbalance = balance // 10`	10
	`balance = ancbalance + montant`	20
	`return ancbalance // 10`	20

3.2 Expliquez le comportement des deux threads pour le scénario suivant :

Thread 1	Thread 2	Balance
`UnTraitement("A")`		0
`FaireUnDepot(10)`		0
`ancbalance = balance // 0`		0
	`UnTraitement("B")`	0
	`FaireUnDepot(10)`	0
	`ancbalance = balance // 0`	0
	`balance = ancbalance + montant`	10
	`return ancbalance // 0`	10
`balance = ancbalance + montant`		10
`Return ancbalance // 0`		10

3.3 Tout en justifiant votre réponse, à partir des deux scénarios « 3.1 » et « 3.2 », pensez vous qu'il faudra introduire des modifications dans le précédent fragment de code ? Si oui, lesquelles ?

3.4 Si vous avez apporté des modifications dans votre fragment de code, tout en développant votre réponse, simulez dans le tableau ci-dessus (à l'image des exemples « 3.1 » et « 3.2 »), le comportement des deux threads dans le cas du second scénario « 3.2 ».

Thread 1	Thread 2	Balance

Exercice 4 (20 points) Le but de cet exercice est d'étudier les différents gestionnaires de mise en forme proposés dans « wxWidgets ».
Chaque réponse doit contenir : la figure, le nom du gestionnaire utilisé, les particularités de ce gestionnaire, et une courte explication sur l'option de ce choix.

4.1 Tout en justifiant votre réponse que va afficher (dessiner la figure) l'exécution du fragment de code suivant :

```
WxBoxSizer1 = new wxBoxSizer(wxHORIZONTAL);
this->SetSizer(WxBoxSizer1);
this->SetAutoLayout(true);

WxButton1 = new wxButton(this, ID_WXBUTTON1, wxT("un"), wxPoint(5,5), wxSize(75,25),\
 0, wxDefaultValidator, wxT("WxButton1"));

WxBoxSizer1->Add(WxButton1,0,wxALIGN_CENTER | wxALL,5);

WxButton2 = new wxButton(this, ID_WXBUTTON2, wxT("deux"), wxPoint(5,40),\
 wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton2"));

WxBoxSizer1->Add(WxButton2,0,wxALIGN_CENTER | wxALL,5);

WxButton3 = new wxButton(this, ID_WXBUTTON3, wxT("trois"), wxPoint(5,75),\
 wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton3"));

WxBoxSizer1->Add(WxButton3,0,wxALIGN_CENTER | wxALL,5);

WxButton4 = new wxButton(this, ID_WXBUTTON4, wxT("quatre"), wxPoint(5,110),\
 wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton4"));

WxBoxSizer1->Add(WxButton4,0,wxALIGN_CENTER | wxALL,5);

WxButton5 = new wxButton(this, ID_WXBUTTON5, wxT("cinq"), wxPoint(5,145),\
 wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton5"));

WxBoxSizer1->Add(WxButton5,0,wxALIGN_CENTER | wxALL,5);
```

4.2 Tout en justifiant votre réponse que va afficher (dessiner la figure) l'exécution du fragment de code suivant :

```
WxBoxSizer1 = new wxBoxSizer(wxVERTICAL);
this->SetSizer(WxBoxSizer1);
this->SetAutoLayout(true);

WxButton1 = new wxButton(this, ID_WXBUTTON1, wxT("un"), wxPoint(5,5), wxSize(75,25),\
 0, wxDefaultValidator, wxT("WxButton1"));

WxBoxSizer1->Add(WxButton1,0,wxALIGN_CENTER | wxALL,5);

WxButton2 = new wxButton(this, ID_WXBUTTON2, wxT("trois"), wxPoint(5,40),\
 wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton2"));

WxBoxSizer1->Add(WxButton2,0,wxALIGN_CENTER | wxALL,5);

WxButton3 = new wxButton(this, ID_WXBUTTON3, wxT("cinq"), wxPoint(5,75),\
 wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton3"));

WxBoxSizer1->Add(WxButton3,0,wxALIGN_CENTER | wxALL,5);

WxButton4 = new wxButton(this, ID_WXBUTTON4, wxT("deux"), wxPoint(5,110),\
 wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton4"));

WxBoxSizer1->Add(WxButton4,0,wxALIGN_CENTER | wxALL,5);

WxButton5 = new wxButton(this, ID_WXBUTTON5, wxT("quatre"), wxPoint(5,145),\
 wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton5"));

WxBoxSizer1->Add(WxButton5,0,wxALIGN_CENTER | wxALL,5);
```

4.3 Tout en justifiant votre réponse que va afficher (dessiner la figure) l'exécution du fragment de code suivant :

```
WxGridSizer1 = new wxGridSizer(2, 2, 0, 0);
this->SetSizer(WxGridSizer1);
this->SetAutoLayout(true);

WxButton1 = new wxButton(this, ID_WXBUTTON1, wxT("un"), wxPoint(5,5), wxSize(75,25),
0, wxDefaultValidator, wxT("WxButton1"));
WxGridSizer1->Add(WxButton1,0,wxALIGN_CENTER | wxALL,5);

WxButton2 = new wxButton(this, ID_WXBUTTON2, wxT("trois"), wxPoint(90,5),
wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton2"));
WxGridSizer1->Add(WxButton2,0,wxALIGN_CENTER | wxALL,5);

WxButton3 = new wxButton(this, ID_WXBUTTON3, wxT("cinq"), wxPoint(5,40),
wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton3"));
WxGridSizer1->Add(WxButton3,0,wxALIGN_CENTER | wxALL,5);

WxButton4 = new wxButton(this, ID_WXBUTTON4, wxT("deux"), wxPoint(90,40),
wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton4"));
WxGridSizer1->Add(WxButton4,0,wxALIGN_CENTER | wxALL,5);

WxButton5 = new wxButton(this, ID_WXBUTTON5, wxT("quatre"), wxPoint(5,75),
wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton5"));
WxGridSizer1->Add(WxButton5,0,wxALIGN_CENTER | wxALL,5);
```

4.4 Tout en justifiant votre réponse que va afficher (dessiner la figure) l'exécution du fragment de code suivant :

```
WxFlexGridSizer1 = new wxFlexGridSizer(2, 2, 0, 0);
this->SetSizer(WxFlexGridSizer1);
this->SetAutoLayout(true);

WxButton1   =   new   wxButton(this,   ID_WXBUTTON1,   wxT("trois"),   wxPoint(5,5),\
 wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton1"));

WxFlexGridSizer1->Add(WxButton1,0,wxALIGN_CENTER | wxALL,5);

WxButton2 = new wxButton(this, ID_WXBUTTON2, wxT("un"), wxPoint(90,5), wxSize(75,25),\
 0, wxDefaultValidator, wxT("WxButton2"));

WxFlexGridSizer1->Add(WxButton2,0,wxALIGN_CENTER | wxALL,5);

WxButton3   =   new   wxButton(this,   ID_WXBUTTON3,   wxT("cinq"),   wxPoint(5,40),\
 wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton3"));

WxFlexGridSizer1->Add(WxButton3,0,wxALIGN_CENTER | wxALL,5);

WxButton4   =   new   wxButton(this,   ID_WXBUTTON4,   wxT("deux"),   wxPoint(90,40),\
 wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton4"));

WxFlexGridSizer1->Add(WxButton4,0,wxALIGN_CENTER | wxALL,5);

WxButton5   =   new   wxButton(this,   ID_WXBUTTON5,   wxT("quatre"),   wxPoint(5,75),\
 wxSize(75,25), 0, wxDefaultValidator, wxT("WxButton5"));

WxFlexGridSizer1->Add(WxButton5,0,wxALIGN_CENTER | wxALL,5);
```

4.5 Quel est la distinction principale entre les deux gestionnaires de mise en forme utilisés dans les questions « 4.3 » et « 4.4 »?

Exercice 5 (13 points) Utilisez le programme ci-dessous pour écrire la fonction C++ « intersection » qui permet de réaliser l'intersection entre deux vecteurs d'éléments préalablement triés où chaque élément est du type « int ». Cette fonction accepte comme arguments deux vecteurs et retourne un vecteur.

```
int main() {
    int tabA[] = {3,6,9,12,18,21};
    int tabB[] = {7,8,9,11,18,21,24,29};

    vector<int> a(tabA,tabA+6);
    vector<int> b(tabB,tabB+8);
    vector<int> c = intersection(a,b);

    copy(c.begin(),c.end(),ostream_iterator<int> (cout," ")); // 9 18 21

    cout << endl;

    return 0;
}
```

Exercice 6 (14 points) Soit le programme suivant qui compile et qui s'exécute correctement :

```
01  #include <vector>
02  #include <iostream>
03
04  using namespace std;
05
06  template <typename iterator>
07  iterator Mystere (iterator beg, iterator end) {
08      if (beg==end) return(end);
09      iterator it=beg;
10      ++it;
11      for ( iterator ap=beg; it!=end; ++it, ++ap) {
12          if (*it==*ap)
13              return(ap);
14      }
15      return(end);
16  }
17
18  int main () {
19      int mesints[] = {10,20,30,30,20,10,10,20};
20      vector<int> monvector (mesints,mesints+8);
21      vector<int>::iterator pos;
22
23      pos=Mystere( monvector.begin(), monvector.end());
24
25      if (pos == monvector.end())
26          cout << "Message -1- à compléter." << endl;
27      else
28          cout << "Message -2- à compléter: " << *pos << endl;
29
30      return(0);
31  }
32
```

6.1 Expliquez le rôle de la fonction « Mystere ».

6.2 Que va afficher le programme en sortie ? (Détaillez brièvement votre réponse)

6.3 Récrivez les messages affichés en sortie figurant aux lignes 26 et 28 du précédent programme pour qu'ils reflètent correctement l'intérêt de la situation rencontrée dans le programme.

Ancien message	Nouveau message
Message -1- à compléter	
Message -2- à compléter	

Bonnes vacances, et bonne continuité !