Trimestre Hiver, 2009

Mohamed Lokbani

IFT1169 – Examen Final –

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

Nom: _____| Prénom(s): _____|

Signature: _____| Code perm: _____|

Date : mardi 21 avril 2009

Durée : 3 heures (de 18h30 à 21h30)

Local : Z-220 ; 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._____/05 (1.1 1.2)

2._____/10 (2.1)

3._____/15 (3.1 3.2 3.3 3.4)

4._____/15 (4.1)

5._____/15 (5.1)

6._____/20 (6.1 6.2 6.3 6.4)

7._____/20 (7.1)

Total:_____/100

Directives officielles

* Interdiction de toute communication verbale pendant l'examen.

* Interdiction de quitter la salle pendant la première heure.

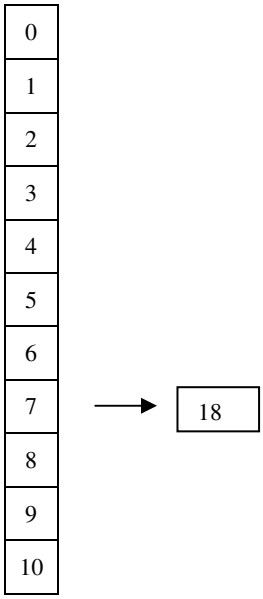
* L'étudiant qui doit s'absenter après la première heure remettra sa carte d'étudiant au surveillant, l'absence ne devant pas dépasser 5 minutes. Un seul étudiant à la fois peut quitter la salle.

* 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 (5 points) Stocker les éléments suivants {18, 17, 39, 51, 34, 73, 50, 22, 79, 45, 66} (dans cet ordre) dans une table de hachage préalablement vide, de taille « 11 », en utilisant la fonction de hachage « `h(x) = x%11` », en vous servant des deux techniques décrites dans les notes de cours.

1.1 Chaînage (buckets)



Description brève de votre démarche

1.2 Première case libre à droite (après)

0	1	2	3	4	5	6	7	8	9	10
							18			

Description brève de votre démarche

Exercice 2 (10 points) Compléter le fichier « makefile » suivant avec toutes les dépendances demandées.

// **Fichier [A.h]** #ifndef CLASS_A #define CLASS_A class A { ... }; #endif	// **Fichier [B.h]** #ifndef CLASS_B #define CLASS_B class B { ... }; #endif	// **Fichier [C.h]** #ifndef CLASS_C #define CLASS_C class C { ... }; #endif
// Fichier [A.cpp] #include "A.h" #include "B.h" // méthodes de la classe A	**// Fichier [B.cpp]** #include "B.h" #include "C.h" // méthodes de la classe B	**// Fichier [C.cpp]** #include "C.h" #include "A.h" // méthodes de la classe C
// Fichier [X.cpp] #include "A.h" #include "B.h" #include "C.h" int main ( ) { ... }	**// Fichier [Y.cpp]** #include "A.h" #include "B.h" #include "C.h" int main ( ) { ... }	**// Fichier [Z.cpp]** #include "A.h" #include "B.h" #include "C.h" int main ( ) { ... }

```
# Makefile
all:  X Y Z

A.o:


B.o:


C.o:


X.o:


Y.o:


Z.o:


X:


Y:


Z:


clean:
      rm *.o
```

Exercice 3 (15 points)

3.1 Comment compiler avec « g++ » un programme en faisant appel à la librairie « pthread »?

3.2 Dans un programme « multithreadés », si un thread crée une variable locale, tous les threads ont-ils accès à cette variable?

3.3 Un serveur crée un thread pour chaque client. Pas plus de N threads (et donc clients) ne peuvent être actives immédiatement. Comment dire au thread « main » qu'un thread enfant s'est terminé et qu'il peut servir maintenant un autre client?

3.4 Tout en ignorant le retour des différentes fonctions (pthread), le fragment de code suivant pose problème. Expliquer cela brièvement et proposer 2 manières pour fixer ce problème.

```
01  pthread_mutex_t mutexA,mutexB;
02
03  void *func1(void *){
04      pthread_mutex_lock(&mutexA);
05      pthread_mutex_lock(&mutexB);
06      // section critique
07      pthread_mutex_unlock(&mutexB);
08      pthread_mutex_unlock(&mutexA);
09  }
10
11  void *func2(void *){
12      pthread_mutex_lock(&mutexB);
13      pthread_mutex_lock(&mutexA);
14      // section critique
15      pthread_mutex_unlock(&mutexA);
16      pthread_mutex_unlock(&mutexB);
17  }
18
19  int main(){
20      pthread_t thread1, thread2 ;
21      pthread_mutex_init(&mutexA, NULL);
22      pthread_mutex_init(&mutexB, NULL);
23      pthread_create(&thread1, NULL, func1, NULL);
24      pthread_create(&thread2, NULL, func2, NULL);
25      pthread_mutex_destroy(&mutexA);
26      pthread_mutex_destroy(&mutexB);
27      return 0;
28  }
```

Exercice 4 (15 points) Que va afficher en sortie le programme suivant qui compile et s'exécute correctement?

```
01  #include <iostream>
02
03  using namespace std;
04
05  class Ex1 {
06      const char *msg;
07  public:
08      Ex1 (const char *m): msg(m) { }
09      const char *affiche( ) const {return msg;}
10  };
11
12  class Ex2 {
13      int valeur;
14  public:
15      Ex2 (int v): valeur(v) { }
16      int getValeur ( ) { return valeur; }
17  };
18
19  class Ex3 { };
20
21  int F (int a, int b) {
22      if (a==0) throw new Ex3;
23      if (a==3) return 10*b;
24      if (a>b) throw new Ex2(a-b);
25      if (a<b) throw new Ex1("Inferieur");
26      return a+b;
27  }
28  int main ( ) {
29      for (int j=-1; j<=3; j++) {
30          for (int k=-1; k<=2; k++) {
31              try { cout << F(j,k); }
32              catch (Ex1 *ex1) { cout << ex1->affiche( ); }
33              catch (Ex2 *ex2) { cout << ex2->getValeur( ); }
34              catch ( ... ) { cout << "Erreur"; }
35              cout << '\t';
36          }
37          cout << endl;
38      }
39      return 0;
49  }
```

Affichage en sortie et description brève de votre démarche

Exercice 5 (15 points) Que va dessiner le fragment de code suivant?

```
01  UnSizer::UnSizer(const wxString& titre)
02          : wxFrame(NULL, -1, titre, wxPoint(-1, -1), wxSize(270, 220))
03  {
04    menubar = new wxMenuBar;
05    file = new wxMenu;
06
07    SetMenuBar(menubar);
08
09    sizer = new wxBoxSizer(wxVERTICAL);
10
11    display = new wxTextCtrl(this, -1, wxT(""), wxPoint(-1, -1),
12       wxSize(-1, -1), wxTE_RIGHT);
13
14    sizer->Add(display, 0, wxEXPAND | wxTOP | wxBOTTOM, 4);
15    gs = new wxGridSizer(4, 4, 3, 3);
16
17    gs->Add(new wxButton(this, -1, wxT("Effacer")), 0, wxEXPAND);
18    gs->Add(new wxStaticText (this, -1, wxT("")), 0, wxEXPAND);
19    gs->Add(new wxStaticText(this, -1, wxT("")), 0, wxEXPAND);
20    gs->Add(new wxButton(this, -1, wxT("Fermer")), 0, wxEXPAND);
21    gs->Add(new wxButton(this, -1, wxT("+")), 0, wxEXPAND);
22    gs->Add(new wxButton(this, -1, wxT("-")), 0, wxEXPAND);
23    gs->Add(new wxButton(this, -1, wxT("*")), 0, wxEXPAND);
24    gs->Add(new wxButton(this, -1, wxT("/")), 0, wxEXPAND);
25    gs->Add(new wxButton(this, -1, wxT("=")), 0, wxEXPAND);
26    gs->Add(new wxButton(this, -1, wxT("3")), 0, wxEXPAND);
27    gs->Add(new wxButton(this, -1, wxT("4")), 0, wxEXPAND);
28    gs->Add(new wxButton(this, -1, wxT("7")), 0, wxEXPAND);
29    gs->Add(new wxButton(this, -1, wxT(".")), 0, wxEXPAND);
30    gs->Add(new wxButton(this, -1, wxT("2")), 0, wxEXPAND);
31    gs->Add(new wxButton(this, -1, wxT("5")), 0, wxEXPAND);
32    gs->Add(new wxButton(this, -1, wxT("8")), 0, wxEXPAND);
33    gs->Add(new wxButton(this, -1, wxT("0")), 0, wxEXPAND);
34    gs->Add(new wxButton(this, -1, wxT("1")), 0, wxEXPAND);
35    gs->Add(new wxButton(this, -1, wxT("6")), 0, wxEXPAND);
36    gs->Add(new wxButton(this, -1, wxT("9")), 0, wxEXPAND);
37
38    sizer->Add(gs, 1, wxEXPAND);
39    SetSizer(sizer);
40    SetMinSize(wxSize(270, 220));
41
42    Centre();
43  }
```

L'argument « titre » est initialisée avec la chaîne « Exercice 6 ».

Description brève de votre démarche

Exercice 6 (20 points)

6.1 Compléter la classe « FileIterator » figurant dans le programme ci-dessus avec la déclaration des différents membres et la définition des différentes méthodes.

```
01  class File {
02      int tete, queue, compteur, data[10];
03  public:
04      File( ) : tete(0), queue(0), compteur(0) { } // Head, Tail
05      bool EstVide( ) const { return compteur == 0; } // IsEmpty
06      bool EstPleine( ) const { return compteur == 10; } // IsFull
07      bool Enfiler(int); // Enqueue
08      bool Defiler(int &); // Dequeue
09      friend class FileIterator;
10  };
11
12  bool File::Enfiler(int e) {
13      if (EstPleine()) return false;
14      data[queue++] = e;
15      queue %= 10;
16      compteur++;
17      return true;
18  }
19
20  bool File::Defiler(int& e) {
21      if (EstVide()) return false;
22      e = data[tete++];
23      tete %= 10;
24      compteur--;
25      return true;
26  }
27
28  // Un itérateur pour la classe « File » (« Queue »)
29  class FileIterator {
30
31
32
33  public:
34      // constructeur permettant d'initialiser les membres privés
35      FileIterator(File& q){
36
37
38
39
49
50     }
51      // Retourne vraie s'il y a des éléments dans la file
52      bool APlusieursElements( ) {
53
54
55
56
57
58      }
59      // Retourne la valeur courante
60      int getValeur( ) {
61
62
63
64
65
66      }
67      // Incrémente l'index
68      void suivant( ) {
69
70
71
72
73      }
74  };
```

6.2 Compléter la boucle « for » (**ligne 18**) de la fonction « main » suivante :

```
01  int main() {
02
03      File t;
04
05      // Ajout de 10 25 39
06
07      if (t.Enfiler(10))
08          cout << "insertion ok de 10\n";
09      if (t.Enfiler(25))
10          cout << "insertion ok de 25\n";
11      if (t.Enfiler(39))
12          cout << "insertion ok de 39\n";
13
14      // Affichage
15      FileIterator qt(t);
16
17      // On affiche toutes les valeurs de la File t
18      for (                                                    )
19
20          cout << qt.getValeur() << endl;
21
22      // Retrait
23
24      int a;
25
26      if (t.Defiler(a))
27          cout << "Retrait ok de: " << a << "\n";
28
29
30      return 0;
31
32  }
```

6.3 Que va afficher en sortie le précédent programme?

6.4 Quel est la nature de cette « File » et pourquoi?

Exercice 7 (20 points)

```
01  #include <deque>
02  #include <iterator>
03  #include <algorithm>
04  #include <numeric>
05  #include <functional>
06  using namespace std;
07
08  void print(int value)
09      { cout << value << endl; }
10
11  int main( ) {
12      deque<int> d;
13      d.push_back(4);
14      d.push_back(3);
15      d.push_back(2);
16      d.push_back(1);
17      print(d.front( ));
18
19      // Affichage Obtenu
20
21
22
23      d.pop_front( );
24      d.push_front(5);
25      d.push_front(10);
26      print(d.back( ));
27
28      // Affichage Obtenu
29
30
31
32      d.pop_back( );
33      cout << endl;
34
35      deque<int>::iterator it;
36
37      for (it=d.begin( ); it!=d.end( ); ++it) print(*it);
38
39      // Affichage Obtenu
40
41
42
43
44
45
46
47
48
49      cout << endl;
50
51      int total=0;
52      for (it=d.begin( ); it!=d.end( ); ++it) total=*it-total;
53      print(total);
54      // Affichage Obtenu
55
56
57
58      print(accumulate(d.begin( ), d.end( ), 0, plus<int>( )));
59      // Affichage Obtenu
60
61
62
63      print(accumulate(d.begin( ), d.end( ), 1, multiplies<int>( )));
64      // Affichage Obtenu
65
66
67
68
```

```
69      cout << endl;
70      d.push_back(7);
71      d.push_front(8);
72      for_each (d.begin( ), d.end( ), print);
73
74      // Affichage Obtenu
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90      return 0;
91  }
```

Description brève de votre démarche

Bon Été!