Trimestre Automne, 2023

Mohamed Lokbani

IFT1169 – Examen Final –

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

Nom : _____| Prénom(s) : _____|

Signature : _____| Login : _____|

Date : jeudi 14 décembre 2023
Durée : 3 heures (de 16h30 à 19h30)
Local : Z-205, Pavillon Claire McNicoll

Directives :

- Toute documentation est permise.
- Appareils électroniques **non** permis.

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

1._____/20 (1.1 à 1.10)

2._____/15 (2.1)

3._____/15 (3.1, 3.2, 3.3)

4._____/15 (4.1)

5._____/20 (5.1)

6._____/15 (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 (20 points) répondez par « vrai » ou « faux » **en y incluant une très courte explication**.

1.1 [VRAI | FAUX] « cout » fait partie de l'espace de nom « Garfield ».

1.2 [VRAI | FAUX] Une directive « using » peut être placée en haut d'un fichier « .cpp » ou à l'intérieur d'une définition de classe ou de fonction.

1.3 [VRAI | FAUX] Si une variable locale a le même nom qu'une variable d'espace de noms, la variable d'espace de noms est masquée.

1.4 [VRAI | FAUX] Un espace de nom ne permet pas de regrouper plusieurs déclarations de variables, fonctions et classes dans un groupe anonyme.

1.5 [VRAI | FAUX] Les espaces de noms ont été introduits pour éviter les conflits de noms.

1.6 [VRAI | FAUX] Les espaces de noms représentent un des paradigmes de la programmation orientée objet.

1.7 [VRAI | FAUX] On peut créer un espace de nom dans une classe.

1.8 [VRAI | FAUX] On peut imbriquer des espaces de noms.

1.9 [VRAI | FAUX] Il est possible de définir plusieurs fois un espace de nom dans un programme donné.

1.10 [VRAI | FAUX] Un espace anonyme est utile pour rendre les déclarations de variables visibles à d'autres fichiers.

Exercice 2 (15 points) t**out en développant votre réponse**, que va afficher en sortie le programme suivant qui s'exécute et compile correctement.

```
 1 #include <string>
 2 #include <iostream>
 3 using namespace std;
 4
 5 class UneException{};
 6 class Exo02{
 7 public:
 8     Exo02(string s):Nom(s) { PrintMsg("Constructeur Exo02:"); }
 9     Exo02(const Exo02& autre):Nom(autre.Nom){
10         PrintMsg("Constructeur de recopie Exo02:"); }
11     ~Exo02(){ PrintMsg("Destroyed Exo02:"); }
12     void PrintMsg(string s) { cout << s  << Nom <<  endl; }
13     string Nom;
14 };
15 void C(Exo02 d, int i){
16     cout << "Entree FunctionC" << endl;
17     d.Nom = " C";
18     throw UneException();
19     cout << "Sortie FunctionC" << endl;
20 }
21 void B(Exo02 d, int i){
22     cout << "Entree FunctionB" << endl;
23     d.Nom = "B";
24     C(d, i + 1);
25     cout << "Sortie FunctionB" << endl;
26 }
27 void A(Exo02 d, int i){
28     cout << "Entree FunctionA" << endl;
29     d.Nom = " A";
30     Exo02* ptra = new Exo02("new Exo02");
31     B(d, i + 1);
32     delete ptra;
33     cout << "Sortie FunctionA" << endl;
34 }
35 int main(){
36     cout << "Entree main" << endl;
37     try{
38         Exo02 d(" M");
39         A(d,1);
40     }
41     catch (UneException& e){
42         cout << "Une exception a ete detectee du type: "\
43                                         << typeid(e).name() << endl;
44     }
45     cout << "Sortie main." << endl;
46     return 0;
47 }
```

Exercice 3 (15 points) vous avez conçu un jeu en ligne. Chaque fois qu'un joueur termine de jouer, vous allez vérifier s'il a fait le meilleur score. Si c'est le cas, vous allez préserver le score et l'identité du joueur. Vous allez avoir grosso modo le code suivant :

```
int max_score; // le score le plus élevé
int max_id; // id du joueur ayant obtenu le score le plus élevé

void maj(int score, int id){
    if (score > max_score) {
        max_score = score;
        max_id = id;
    }
}

void affiche(){
    cout << "max score est : " << max_score << endl;
    cout << "par le joueur : " << max_id << endl;
}
```

Nous allons examiner ce fragment de code dans le cas de deux joueurs qui jouent en même temps. Nous allons supposer que le système permet la gestion de threads. Chaque thread va exécuter ces tâches de manières concurrentes (bref la notion de threads).

On vous demande de nous expliquer si les situations ci-dessous posent des problèmes. Est-ce que les appels aux fonctions vont fonctionner correctement. Dans tous les cas, on vous demande de justifier votre réponse.

3.1 Les deux threads exécutent la fonction « maj » pour mettre à jour le score de deux différents joueurs.

3.2 Le premier thread exécute la fonction « maj » et le second thread exécute la fonction « affiche ».

3.3 Les deux threads exécutent la fonction « affiche ».

Exercice 4 (15 points) expliquer les lignes du programme suivant et dessiner le résultat obtenu en sortie. Le programme compile et s'exécute sans erreurs.

```
01 #include <wx/wx.h>
02
03 class wxMiniApp : public wxApp{
04 public:
05      virtual bool OnInit();
06
07      void OnClick(wxCommandEvent& event) {
08           GetTopWindow()->Close();
09      }
10 };
11
12 IMPLEMENT_APP(wxMiniApp);
13
14 bool wxMiniApp::OnInit(){
15      SetTopWindow( new wxFrame( NULL, -1, wxT("A23Exo04"),\
16           wxDefaultPosition, wxSize( 200, 150) ) );
17
18      new wxButton( GetTopWindow(), wxID_EXIT, wxT("Click!") );
19
20      Connect(wxID_EXIT, wxEVT_COMMAND_BUTTON_CLICKED, \
21           wxCommandEventHandler(wxMiniApp::OnClick) );
22
23      GetTopWindow()->Show();
24      return true;
25 }
```

Exercice 5 (20 points) que va afficher en sortie le programme suivant qui compile et s'exécute correctement

```
01  #include <iostream>
02  #include <deque>
03  #include <iterator>
04  #include <algorithm>
05  #include <numeric>
06  #include <functional>
07
08  using namespace std;
09
10  void print(int value){
11     cout << value << endl;
12  }
13  int main() {
14     deque<int> d;
15     d.push_back(4);
16     d.push_back(3);
17     d.push_back(2);
18     d.push_back(1);
19     print(d.front());
20     d.pop_front();
21     d.push_front(5);
22     d.push_front(10);
23     print(d.back());
24     d.pop_back();
25     cout << "----------" << endl;
26     deque<int>::iterator it;
27     for (it=d.begin(); it!=d.end(); ++it) print(*it);
28     cout << "----------" << endl;
29     int total=0;
30     for (it=d.begin(); it!=d.end(); ++it) total=*it-total;
31     print(total);
32     print(accumulate(d.begin(), d.end(), 0, plus<int>()));
33     print(accumulate(d.begin(), d.end(), 1, multiplies<int>()));
34     cout << "----------" << endl;
35     d.push_back(7);
36     d.push_front(8);
37     for_each (d.begin(), d.end(), print);
38
39     return 0;
40  }
```

1- « accumulate(arg1,arg2,arg3) » initialise le résultat avec la valeur du arg3 et en ajoute la valeur de chacun des éléments de l'ensemble défini dans l'intervalle [arg1,arg2). C'est une version généralisée de la sommation.
2- « accumulate(arg1,arg2,arg3,xyz) » le principe reste le même, sauf qu'au lieu d'utiliser la version généralisée de la sommation, nous utilisons la fonction « xyz » passée comme 4^{e} argument.

Exercice 6 (15 points) la fonction « main » utilise la fonction « intersection » pour calculer l'intersection entre les deux vecteurs « tabA » et « tabB ». Chaque élément du vecteur est du type « int » et chaque vecteur est préalablement trié. Écrire la fonction « intersection » qui accepte comme arguments deux vecteurs et retourne un vecteur.

L'utilisation de la fonction « std::set_intersection » n'est pas autorisée.

```
01  #include <iostream>
02  #include <vector>
03  #include <iterator>
04
05  using namespace std;
06
07
08  int main() {
09    int tabA[] = {3,6,9,12,18,21};
10    int tabB[] = {7,8,9,11,18,21,24,29};
11    vector<int> a(tabA,tabA+6);
12    vector<int> b(tabB,tabB+8);
13    vector<int> c = intersection(a,b);
14    copy(c.begin(),c.end(),ostream_iterator<int> (cout," ")); // 9 18 21
15    cout << endl;
16    return 0;
17  }
```

Joyeuses Fêtes et Bonne Année!