

Commencé le mardi 25 avril 2023, 09:30

État Terminé

Terminé le mardi 25 avril 2023, 12:00

Temps mis 2 heures 29 min

Note 10,25 sur 20,00 (51,25%)

Description

Indiquez l'ordre de complexité de temps des algorithmes ci-dessous. On considère l'ordre moyen et/ou amorti. S'il y a plusieurs possibilités, indiquez l'ordre qui est le plus petit possible.

La fonction f a un temps en $O(1)$ et `range` n'ajoute pas de complexité par rapport à une boucle faite à la main.

Question 1

Correct

Note de 0,75
sur 0,75

```
list<int> v;  
auto it = v.begin();  
for (int i : range(n)) {  
    v.insert(it, f(i));  
    --it;  
    v.insert(it, f(-i));  
}
```

Est $O($ $)$ ✓



Question **2**

Correct

Note de 0,75
sur 0,75

```
map<int,int> v;  
for (int i : range(n))  
    v[f(i)] = i;
```

Est $O($  $)$

Question **3**

Incorrect

Note de 0,00
sur 0,75

```
set<int> v;  
for (int i : range(n))  
    v.insert(f(i));
```


Est $O($  $)$

Question **4**

Correct

Note de 1,00
sur 1,00

Choisissez dans la liste **TOUTES** les affirmations **VRAIES** au sujet de l'architecture modèle-vue-controlleur

- ☐ a. Le modèle doit connaître les contrôleurs.
- ☐ b. La vue doit continuellement vérifier l'état du modèle en cas de changement.
- ☒ c. Le modèle doit fournir les méthodes nécessaires pour manipuler ses données. 



Soit le code suivant qui représente une situation d'étudiants qui suivent un cours en ligne. Le code ne compile bien sûr pas et des commentaires sont mis à certains endroit pour remplacer le code fonctionnel. Ce code est la base pour la question qui suit.



```
1. struct ErreurCoursEnLigne : std::logic_error {  
2.     using logic_error::logic_error;  
3. };  
4.  
5. struct ErreurIntraTropDifficile : ErreurCoursEnLigne {  
6.     using ErreurCoursEnLigne::ErreurCoursEnLigne;  
7. };  
8.
```



```

9. struct ErreurRienCompris : ErreurIntraTropDifficile {
10.     using ErreurIntraTropDifficile::ErreurIntraTropDifficile;
11. };
12.
13. struct ErreurPasAssezÉtudié : ErreurIntraTropDifficile {
14.     using ErreurIntraTropDifficile::ErreurIntraTropDifficile;
15. };
16.
17. struct ErreurTwitch : ErreurCoursEnLigne {
18.     using ErreurCoursEnLigne::ErreurCoursEnLigne;
19. };
20.
21. struct ErreurColocsDuChargéOntRebranchéLeRouteur : ErreurTwitch {
22.     using ErreurTwitch::ErreurTwitch;
23. };
24.
25.
26. class Étudiant {
27. public:
28.     void étudier(bool pourVrai) {
29.         if (pourVrai) {
30.             // Refaire ses TD par soi-même et refaire les exemples faits en classe. :D
31.             prêtPourExamen_ = true;
32.         } else {
33.             // Meh, relire les notes de cours une fois la veille de l'exam. :(
34.             prêtPourExamen_ = false;
35.         }
36.     }
37.
38.     void assisterAuCours() {
39.         // Assister au stream sur Twitch.
40.         bool coursFini = false;
41.         while (not coursFini) {
42.             if (not streamTwitchActif())
43.                 throw ErreurColocsDuChargéOntRebranchéLeRouteur("Ah come on!");
44.
45.             coursFini = /* Est-ce que Le chargé a fini de monologuer et/ou commence à avoir faim? */;
46.         }
47.     }
48.
49.     void faireIntra() {
50.         double note = /* Est-ce que ça c'est bien passé? */;
51.         if ( note < 10/20.0 ) {
52.             if (not prêtPourExamen_)
53.                 throw ErreurPasAssezÉtudié("Oups! Les questions demandaient de comprendre...");
54.             else

```



```
55.         throw ErreurRienCompris("De toute façon, les examens sont juste des constructions  
        sociales.");  
56.     } else if (note <= 15/20.0) {  
57.         cout << "Pas pire, pas pire." << "\n";  
58.     } else if (note <= 20/20.0) {  
59.         cout << "Yay!" << "\n";  
60.     }  
61. }  
62.  
63. private:  
64.     bool prêtPourExamen_  
65. };
```

La situation présentée relève d'une oeuvre de fiction. Toute ressemblance avec des situations existantes ou ayant existé est une coïncidence.



Question 5

Correct

Note de 1,50
sur 1,50

Soit le code suivant :

```
1. int main () {
2.     Étudiant marilynPothier;
3.     try {
4.         marilynPothier.assisterAuCours();
5.         marilynPothier.étudier(false);
6.         marilynPothier.faireIntra();
7.     } catch (ErreurPasAssezÉtudié& e) {
8.         // Catch 1
9.     } catch (ErreurRienCompris& e) {
10.        // Catch 2
11.    } catch (ErreurIntraTropDifficile& e) {
12.        // Catch 3
13.    } catch (ErreurCoursEnLigne& e) {
14.        // Catch 4
15.    }
16.
17.    // Fin du programme
18. }
```

Qu'arrive-t-il si le stream reste actif et que l'élève obtient une note de 14/20?

Veuillez choisir une réponse.

- ☐ a. `faireIntra()` affiche *Pas pire, pas pire*, Le catch 1 est exécuté en raison du manque d'étude, puis le programme se termine normalement.
- ☒ b. `faireIntra()` affiche *Pas pire, pas pire*, aucune exception n'est levée, puis le programme se termine normalement. ✓
- ☐ c. `faireIntra()` affiche *Pas pire, pas pire*, Le catch 1 est exécuté en raison du manque d'étude, puis le programme plante en raison de l'exception.



Question 6

Correct

Note de 2,00
sur 2,00

Cette question devrait être faite sans utiliser un compilateur.

On désire déterminer l'ordre de construction des différents objets/sous-objets pour une déclaration donnée. On ne listera pas dans cet ordre la "construction" des types fondamentaux, qui n'ont pas de constructeur.

Soit les définitions de classes suivantes (dans des fichiers séparés et on suppose que les "include" sont faits correctement pour que ça compile):

```
class A : public B, public C {
```

```
public:
```

```
    A() { }
```

```
private:
```

```
    C att1_;
```

```
    B* att2_;
```

```
};
```

```
class B {
```

```
public:
```

```
    B() { }
```

```
};
```

```
class C {
```

```
public:
```

```
    C() { }
```

```
};
```

```
class D : public C {
```

```
public:
```

```
    D() { att2_ = new A(); }
```

```
private:
```

```
    B att1_;
```

```
    A* att2_;
```

```
};
```

Soit le programme suivant:

```
int main() {
```

```
    D x;
```

```
}
```

On veut savoir l'ordre de construction lorsqu'on exécute ce programme. Indice: il y a 7 objets/sous-objets construits.



Indiquez uniquement les noms des types dans le bon ordre, exemple: A B C D A B C

Réponse : (régime de pénalités : 0 %)

Indiquez l'ordre de début de construction pour les classes ci-haut:

D C B A B C C

Indiquez l'ordre de début d'exécution du corps des constructeurs pour les classes ci-haut:

C B D B C C A

Votre réponse	Devrait être	
D C B A B C C	D C B A B C C	✓
C B D B C C A	C B D B C C A	✓

Réponse bien enregistrée: DCBABCC, CBDBCCA

Tous les tests ont été réussis ! ✓

Correct

Note pour cet envoi : 2,00/2,00.



Soient les classes suivantes

```
class Personne {  
public:  
    Personne(string nom) : nom_(nom) {}  
    string getNom() { return nom_; }  
    string getRole(){ return "aucun"; }  
private:  
    string nom_;  
};
```

```
class Arbitre {  
public:  
    Arbitre(string nom, string role) : Personne(nom), role_(role) {  
        nbMatchArbitre_ = Aleatoire(10, 50)();  
    }  
    string getRole() { return role_; }  
    int getNbMatchArbitre() { return nbMatchArbitre_; }  
    void setNbMatchArbitre(int nb) { nbMatchArbitre_ = nb; }  
private:  
    string role_;  
    int nbMatchArbitre_;  
};
```

Question **7**

Correct

Note de 0,50
sur 0,50

Que représente **Aleatoire**, utilisé dans le constructeur de la classe **Arbitre** ?

Veuillez choisir une réponse.

- ☐ 1. une fonction générique qui retourne une valeur aléatoire
- ☐ 2. une méthode de la classe Arbitre
- ☐ 3. un opérateur de la classe Arbitre
- ☐ 4. une fonction globale qui retourne une valeur aléatoire
- ☒ 5. une classe foncteur ✓

Question **8**

Incorrect

Note de 0,00
sur 1,00

Indiquer tous les changements nécessaires dans les deux classes, afin de faire du polymorphisme sur la méthode `getRole()`.

Veuillez choisir au moins une réponse.

- ☒ 1. `class Arbitre : public Personne` ✓
- ☐ 2. `virtual string getRole() { return "aucun"; }`
- ☒ 3. `virtual string getRole() { return role_; }` ✗
- ☐ 4. Dans la classe Arbitre:
 `private:`
 `string nom_;`
 `string role_;`
- ☒ 5. Dans la classe Personne: ✗
 `private:`
 `string nom_;`
 `string role_;`
- ☐ 6. `virtual string getNom() { return nom_; }`



Question 9

Correct

Note de 1,00
sur 1,00

Identifier les éléments adéquats manquants afin d'appliquer le polymorphisme sur la méthode `getRole()`. Vous n'avez pas à remplir les "...", c'est du code qui fait déjà correctement ce qui est indiqué, mais qui est omis de la question car non nécessaire à la compréhension du polymorphisme.

```
set<  ✓ , comparerPersonne>  ✓ listePersonne;  
listePersonne.insert(  ✓ ); // Ajouter une Arbitre  
// ... d'autres Personne et Arbitre sont ajoutés  
for (  ✓ :  ✓ ) { // Parcourir listePersonne  
    cout << // ... affiche le nom  
        << " a le role " <<  ✓ ;  
    if (  ✓ ) // Afficher le nombre de matchs arbitrés par  
l'arbitre  
        cout << " a arbitré " <<  ✓ << endl;  
    else  
        cout << endl;  
}
```



Question **10**

Correct

Note de 1,50
sur 1,50

Écrire le foncteur prédicat binaire **comparerPersonne**, utilisable comme deuxième paramètre d'un conteneur **set** du bon type, qui reçoit comme paramètres deux pointeurs à des objets **Personne** et retourne le résultat de la comparaison selon l'ordre alphabétique croissant des noms des personnes.

Par exemple:

Test	Résultat
Personne simone{"Simone"}, albert{"Albert"}; comparerPersonne{}(&simone, &albert); cout << "ok";	ok
Personne simone{"Simone"}, albert{"Albert"}, marc{"Marc"}, sophie{"Sophie"}; cout << comparerPersonne{}(&simone, &albert) << comparerPersonne{}(&albert, &marc) << comparerPersonne{}(&sophie, &marc) << comparerPersonne{}(&simone, &sophie);	0101

Réponse : (régime de pénalités : 0 %)

```
1 struct comparerPersonne
2 {
3     bool operator()(Personne* a, Personne* b)
4     {
5         //for (int i : range(max{a->getNom(),}))
6         std::string noma(a->getNom());
7         char& f1 = noma.front();
8         std::string nomb(b->getNom());
9         char& f2 = nomb.front();
10        return (f1<=f2);
11    }
12 };
```



	Test	Résultat attendu	Résultat obtenu	
✓	<pre> Personne simone{"Simone"}, albert{"Albert"}; comparerPersonne{ }(&simone, &albert); cout << "ok"; </pre>	ok	ok	✓
✓	<pre> Personne simone{"Simone"}, albert{"Albert"}, marc{"Marc"}, sophie{"Sophie"}; cout << comparerPersonne{ }(&simone, &albert) << comparerPersonne{ }(&albert, &marc) << comparerPersonne{ }(&sophie, &marc) << comparerPersonne{ }(&simone, &sophie); </pre>	0101	0101	✓

Tous les tests ont été réussis ! ✓

Correct

Note pour cet envoi : 1,50/1,50.



Question 11

Correct

Note de 1,25
sur 1,25

Soient les classes **Personne** et **Arbitre** des questions précédentes. Si on voulait que **Professeur** soit une **Personne**, et qu'un **ArbitreProfesseur** soit à la fois un **Arbitre** (qui est une **Personne**) et un **Professeur** mais ne soit qu'une seule **Personne**. Que doit-on minimalement faire?

- ☒ a. Le constructeur de ArbitreProfesseur doit indiquer directement la construction de Personne ✓
- ☐ b. ArbitreProfesseur doit utiliser l'héritage **protected** pour ses classes de base
- ☐ c. ArbitreProfesseur doit avoir comme classe de base directe Personne
- ☒ d. Arbitre doit avoir Personne comme classe de base avec le mot clé **virtual** ✓
- ☐ e. ArbitreProfesseur doit avoir les classes de base Arbitre et Professeur avec le mot clé **virtual**
- ☒ f. Professeur doit avoir Personne comme classe de base avec le mot clé **virtual** ✓



Soient les classes Arbitre (sans polymorphisme, contrairement à la section précédente) et GestionnaireArbitre. L'attribut arbitres_ contient tous les arbitres quel que soit leur rôle. L'attribut map filtreArbitreRole_ a pour cle le rôle de l'arbitre, et comme valeur tous les arbitres qui ont ce rôle.

```
class Arbitre {
public:
    Arbitre (string nom, string role) : nom_(nom), role_(role){
        nbMatchArbitre_ = Aleatoire(10, 50)();
    }
    string getNom() const { return nom_; }
    string getRole() const { return role_; }
    int getNbMatchArbitre() const { return nbMatchArbitre_; }
    void setNbMatchArbitre(int nb) { nbMatchArbitre_ = nb; }
private:
    string nom_;
    string role_;
    int nbMatchArbitre_;
};

class GestionnaireArbitre {
public:
    GestionnaireArbitre() = default;
    bool ajouterArbitre(Arbitre* a);
    bool supprimerArbitre(string nom);
    Arbitre* getArbitreParNom(string nom) const;
    friend ostream& operator << (ostream& sortie,
        const GestionnaireArbitre& g);
protected:
    vector<Arbitre*>::const_iterator getIteratorArbitreParNom(string nom) const;
private:
    vector<Arbitre*> arbitres_;
    unordered_map<string, vector<Arbitre*>> filtreArbitreRole_;
};
```

Question 12

Incorrect

Note de 0,00
sur 1,00

La fonction membre protégée `getIteratorArbitreParNom` retourne l'itérateur const vers l'arbitre du conteneur `arbitres_` qui a le nom spécifié en paramètre. En utilisant la STL, sans `for/while`, écrire en une ligne le corps de la fonction.

Par exemple:

Test	Résultat
<pre>GestionnaireArbitre gestionArbitre; gestionArbitre.ajouterArbitre(new Arbitre("albert","principal")); gestionArbitre.ajouterArbitre(new Arbitre("albert","secondaire")); cout << gestionArbitre;</pre>	<pre>le contenu de gestionnaire des arbitres Le conteneur Vector albert principal Le conteneur Map principal albert</pre>

Réponse : (régime de pénalités : 0, 0, 0, 0, 0, 3 %)

Réinitialiser la réponse

```
1 vector<Arbitre*>::const_iterator GestionnaireArbitre::getIt  
2     auto it = find(arbitres_.begin(), arbitres_.end(), [nom  
3     return it;  
4
```

Erreur(s) de syntaxe




```

tests.cpp: In member function 'std::vector<Arbitre*>::const_iterator
GestionnaireArbitre::getIteratorArbitreParNom(std::__cxx11::string) const':
tests.cpp:4:47: error: qualified-id in declaration before '(' token
tests.cpp:4:55: error: expected primary-expression before 'nom'
tests.cpp:11:41: error: qualified-id in declaration before '(' token
tests.cpp:21:43: error: qualified-id in declaration before '(' token
tests.cpp:43:1: error: a function-definition is not allowed here before '{' token
tests.cpp:64:12: error: a function-definition is not allowed here before '{' token
tests.cpp:83:1: error: expected '}' at end of input
In file included from /usr/include/c++/7/bits/stl_algobase.h:71:0,
                 from /usr/include/c++/7/bits/char_traits.h:39,
                 from /usr/include/c++/7/ios:40,
                 from /usr/include/c++/7/ostream:38,
                 from /usr/include/c++/7/iostream:39,
                 from ./headers_getIteratorArbitreParNom.hpp:1,
                 from <command-line>:0:
/usr/include/c++/7/bits/predefined_ops.h: In instantiation of 'bool
__gnu_cxx::__ops::_Iter_equals_val<_Value>::operator()(_Iterator) [with _Iterator =
__gnu_cxx::__normal_iterator<Arbitre* const*, std::vector<Arbitre*> >; _Value = const
GestionnaireArbitre::getIteratorArbitreParNom(std::__cxx11::string) const::<lambda(Arbitre*)>]':
/usr/include/c++/7/bits/stl_algo.h:120:14:   required from '_RandomAccessIterator
std::__find_if(_RandomAccessIterator, _RandomAccessIterator, _Predicate, std::random_access_iterator_tag) [with
_RandomAccessIterator = __gnu_cxx::__normal_iterator<Arbitre* const*, std::vector<Arbitre*> >; _Predicate =
__gnu_cxx::__ops::_Iter_equals_val<const GestionnaireArbitre::getIteratorArbitreParNom(std::__cxx11::string)
const::<lambda(Arbitre*)> >]'
/usr/include/c++/7/bits/stl_algo.h:161:23:   required from '_Iterator std::__find_if(_Iterator, _Iterator,
_Predicate) [with _Iterator = __gnu_cxx::__normal_iterator<Arbitre* const*, std::vector<Arbitre*> >; _Predicate
= __gnu_cxx::__ops::_Iter_equals_val<const GestionnaireArbitre::getIteratorArbitreParNom(std::__cxx11::string)
const::<lambda(Arbitre*)> >]'
/usr/include/c++/7/bits/stl_algo.h:3907:28:   required from '_IIter std::find(_IIter, _IIter, const _Tp&) [with
_IIter = __gnu_cxx::__normal_iterator<Arbitre* const*, std::vector<Arbitre*> >; _Tp =
GestionnaireArbitre::getIteratorArbitreParNom(std::__cxx11::string) const::<lambda(Arbitre*)>]
__tester__.cpp:2:101:   required from here
/usr/include/c++/7/bits/predefined_ops.h:241:17: error: no match for 'operator==' (operand types are 'Arbitre*
const' and 'const GestionnaireArbitre::getIteratorArbitreParNom(std::__cxx11::string) const::
<lambda(Arbitre*)>')
    { return *__it == _M_value; }
           ~~~~~^~~~~~
In file included from /usr/include/c++/7/bits/stl_algobase.h:67:0,
                 from /usr/include/c++/7/bits/char_traits.h:39,
                 from /usr/include/c++/7/ios:40,

```



```

        from /usr/include/c++/7/ostream:38,
        from /usr/include/c++/7/iostream:39,
        from ./headers_getIteratorArbitreParNom.hpp:1,
        from <command-line>:0:
/usr/include/c++/7/bits/stl_iterator.h:862:5: note: candidate: template<class _IteratorL, class _IteratorR,
class _Container> bool __gnu_cxx::operator==(const __gnu_cxx::__normal_iterator<_IteratorL, _Container>&, const
__gnu_cxx::__normal_iterator<_IteratorR, _Container>&)
    operator==(const __normal_iterator<_IteratorL, _Container>& __lhs,
    ~~~~~~
/usr/include/c++/7/bits/stl_iterator.h:862:5: note: template argument deduction/substitution failed:
In file included from /usr/include/c++/7/bits/stl_algobase.h:71:0,
        from /usr/include/c++/7/bits/char_traits.h:39,
        from /usr/include/c++/7/ios:40,
        from /usr/include/c++/7/ostream:38,
        from /usr/include/c++/7/iostream:39,
        from ./headers_getIteratorArbitreParNom.hpp:1,
        from <command-line>:0:
/usr/include/c++/7/bits/predefined_ops.h:241:17: note: mismatched types 'const
__gnu_cxx::__normal_iterator<_IteratorL, _Container>' and 'Arbitre* const'
    { return *__it == _M_value; }
    ~~~~~^~~~~~
In file included from /usr/include/c++/7/bits/stl_algobase.h:67:0,
        from /usr/include/c++/7/bits/char_traits.h:39,
        from /usr/include/c++/7/ios:40,
        from /usr/include/c++/7/ostream:38,
        from /usr/include/c++/7/iostream:39,
        from ./headers_getIteratorArbitreParNom.hpp:1,
        from <command-line>:0:
/usr/include/c++/7/bits/stl_iterator.h:869:5: note: candidate: template<class _Iterator, class _Container> bool
__gnu_cxx::operator==(const __gnu_cxx::__normal_iterator<_Iterator, _Container>&, const
__gnu_cxx::__normal_iterator<_Iterator, _Container>&)
    operator==(const __normal_iterator<_Iterator, _Container>& __lhs,
    ~~~~~~
/usr/include/c++/7/bits/stl_iterator.h:869:5: note: template argument deduction/substitution failed:
In file included from /usr/include/c++/7/bits/stl_algobase.h:71:0,
        from /usr/include/c++/7/bits/char_traits.h:39,
        from /usr/include/c++/7/ios:40,
        from /usr/include/c++/7/ostream:38,
        from /usr/include/c++/7/iostream:39,
        from ./headers_getIteratorArbitreParNom.hpp:1,
        from <command-line>:0:

```



```

/usr/include/c++/7/bits/predefined_ops.h:241:17: note:   mismatched types 'const
__gnu_cxx::__normal_iterator<_Iterator, _Container>' and 'Arbitre* const'
{ return *__it == _M_value; }
      ~~~~~^~~~~~
In file included from /usr/include/x86_64-linux-gnu/c++/7/bits/c++allocator.h:33:0,
      from /usr/include/c++/7/bits/allocator.h:46,
      from /usr/include/c++/7/string:41,
      from /usr/include/c++/7/bits/locale_classes.h:40,
      from /usr/include/c++/7/bits/ios_base.h:41,
      from /usr/include/c++/7/ios:42,
      from /usr/include/c++/7/ostream:38,
      from /usr/include/c++/7/iostream:39,
      from ./headers_getIteratorArbitreParNom.hpp:1,
      from <command-line>:0:
/usr/include/c++/7/ext/new_allocator.h:155:5: note: candidate: template<class _Tp> bool __gnu_cxx::operator==
(const __gnu_cxx::new_allocator<_Tp>&, const __gnu_cxx::new_allocator<_Tp>&)
operator==(const new_allocator<_Tp>&, const new_allocator<_Tp>&)
      ^~~~~~
/usr/include/c++/7/ext/new_allocator.h:155:5: note:   template argument deduction/substitution failed:
In file included from /usr/include/c++/7/bits/stl_algobase.h:71:0,
      from /usr/include/c++/7/bits/char_traits.h:39,
      from /usr/include/c++/7/ios:40,
      from /usr/include/c++/7/ostream:38,
      from /usr/include/c++/7/iostream:39,
      from ./headers_getIteratorArbitreParNom.hpp:1,
      from <command-line>:0:
/usr/include/c++/7/bits/predefined_ops.h:241:17: note:   mismatched types 'const __gnu_cxx::new_allocator<_Tp>'
and 'Arbitre* const'
{ return *__it == _M_value; }
      ~~~~~^~~~~~

```

Incorrect

Note pour cet envoi : 0,00/1,00.



Question **13**

Incorrect

Note de 0,00
sur 1,00

La fonction membre **getArbitreParNom** retourne le pointeur vers l'arbitre qui a le nom spécifié en paramètre, ou **nullptr** s'il n'y a pas d'arbitre de ce nom. Écrire le corps de cette fonction proprement en évitant la duplication de code.

Par exemple:

Test	Résultat
<pre>GestionnaireArbitre gestionArbitre; gestionArbitre.ajouterArbitre(new Arbitre("albert","principal")); gestionArbitre.ajouterArbitre(new Arbitre("albert","secondaire")); cout << gestionArbitre;</pre>	<pre>le contenu de gestionnaire des arbitres Le conteneur Vector albert principal Le conteneur Map principal albert</pre>

Réponse : (régime de pénalités : 0, 0, 0, 0, 0, 3 %)

Réinitialiser la réponse

```
1 Arbitre* GestionnaireArbitre::getArbitreParNom(string nom) co  
2 {  
3     if(auto it = find(arbitres_.begin(), arbitres_.end(), [no  
4     {  
5         return nullptr;  
6     }  
7     else  
8     {  
9         return arbitres_[it];  
10    }  
11 }
```



Erreur(s) de syntaxe



```

__tester__.cpp: In member function 'Arbitre* GestionnaireArbitre::getArbitreParNom(std::__cxx11::string)
const':
__tester__.cpp:3:110: error: could not convert '(it = ((const GestionnaireArbitre*)this)-
>GestionnaireArbitre::arbitres_.std::vector<Arbitre*>::end())' from '__gnu_cxx::__normal_iterator<Arbitre*
const*, std::vector<Arbitre*> >' to 'bool'
    if(auto it = find(arbitres_.begin(), arbitres_.end(), [nom](Arbitre* x){return (x->getNom()==nom);}); it =
arbitres_.end())

~~~~~^~~~~~
__tester__.cpp:9:25: error: no match for 'operator[]' (operand types are 'const std::vector<Arbitre*>' and
 '__gnu_cxx::__normal_iterator<Arbitre* const*, std::vector<Arbitre*> >')
    return arbitres_[it];
                   ^

In file included from /usr/include/c++/7/vector:64:0,
                 from ./headers_getIteratorArbitreParNom.hpp:5,
                 from <command-line>:0:
/usr/include/c++/7/bits/stl_vector.h:795:7: note: candidate: std::vector<_Tp, _Alloc>::reference
std::vector<_Tp, _Alloc>::operator[](std::vector<_Tp, _Alloc>::size_type) [with _Tp = Arbitre*; _Alloc =
std::allocator<Arbitre*>; std::vector<_Tp, _Alloc>::reference = Arbitre*&; std::vector<_Tp, _Alloc>::size_type
= long unsigned int]
    operator[](size_type __n) _GLIBCXX_NOEXCEPT
    ^~~~~~
/usr/include/c++/7/bits/stl_vector.h:795:7: note: no known conversion for argument 1 from
 '__gnu_cxx::__normal_iterator<Arbitre* const*, std::vector<Arbitre*> >' to 'std::vector<Arbitre*>::size_type
{aka long unsigned int}'
/usr/include/c++/7/bits/stl_vector.h:813:7: note: candidate: std::vector<_Tp, _Alloc>::const_reference
std::vector<_Tp, _Alloc>::operator[](std::vector<_Tp, _Alloc>::size_type) const [with _Tp = Arbitre*; _Alloc =
std::allocator<Arbitre*>; std::vector<_Tp, _Alloc>::const_reference = Arbitre* const&; std::vector<_Tp,
_Alloc>::size_type = long unsigned int]
    operator[](size_type __n) const _GLIBCXX_NOEXCEPT
    ^~~~~~
/usr/include/c++/7/bits/stl_vector.h:813:7: note: no known conversion for argument 1 from
 '__gnu_cxx::__normal_iterator<Arbitre* const*, std::vector<Arbitre*> >' to 'std::vector<Arbitre*>::size_type
{aka long unsigned int}'
In file included from /usr/include/c++/7/bits/stl_algobase.h:71:0,
                 from /usr/include/c++/7/bits/char_traits.h:39,
                 from /usr/include/c++/7/ios:40,
                 from /usr/include/c++/7/ostream:38,
                 from /usr/include/c++/7/iostream:39,
                 from ./headers_getIteratorArbitreParNom.hpp:1,
                 from <command-line>:0:

```



```

/usr/include/c++/7/bits/predefined_ops.h: In instantiation of 'bool
__gnu_cxx::__ops::_Iter_equals_val<_Value>::operator()(_Iterator) [with _Iterator =
__gnu_cxx::__normal_iterator<Arbitre* const*, std::vector<Arbitre*> >; _Value = const
GestionnaireArbitre::getArbitreParNom(std::__cxx11::string) const::<lambda(Arbitre*)>]':
/usr/include/c++/7/bits/stl_algo.h:120:14:   required from '_RandomAccessIterator
std::__find_if(_RandomAccessIterator, _RandomAccessIterator, _Predicate, std::random_access_iterator_tag) [with
_RandomAccessIterator = __gnu_cxx::__normal_iterator<Arbitre* const*, std::vector<Arbitre*> >; _Predicate =
__gnu_cxx::__ops::_Iter_equals_val<const GestionnaireArbitre::getArbitreParNom(std::__cxx11::string) const::
<lambda(Arbitre*)> >]'
/usr/include/c++/7/bits/stl_algo.h:161:23:   required from '_Iterator std::__find_if(_Iterator, _Iterator,
_Predicate) [with _Iterator = __gnu_cxx::__normal_iterator<Arbitre* const*, std::vector<Arbitre*> >; _Predicate =
__gnu_cxx::__ops::_Iter_equals_val<const GestionnaireArbitre::getArbitreParNom(std::__cxx11::string) const::
<lambda(Arbitre*)> >]'
/usr/include/c++/7/bits/stl_algo.h:3907:28:   required from '_IIter std::find(_IIter, _IIter, const _Tp&) [with
_IIter = __gnu_cxx::__normal_iterator<Arbitre* const*, std::vector<Arbitre*> >; _Tp =
GestionnaireArbitre::getArbitreParNom(std::__cxx11::string) const::<lambda(Arbitre*)>]'
__tester__.cpp:3:104:   required from here
/usr/include/c++/7/bits/predefined_ops.h:241:17: error: no match for 'operator==' (operand types are 'Arbitre*
const' and 'const GestionnaireArbitre::getArbitreParNom(std::__cxx11::string) const::<lambda(Arbitre*)>')
    { return *__it == _M_value; }
          ~~~~~^~~~~~

```

```

In file included from /usr/include/c++/7/bits/stl_algobase.h:67:0,
    from /usr/include/c++/7/bits/char_traits.h:39,
    from /usr/include/c++/7/ios:40,
    from /usr/include/c++/7/ostream:38,
    from /usr/include/c++/7/iostream:39,
    from ./headers_getIteratorArbitreParNom.hpp:1,
    from <command-line>:0:

```

```

/usr/include/c++/7/bits/stl_iterator.h:862:5: note: candidate: template<class _IteratorL, class _IteratorR,
class _Container> bool __gnu_cxx::operator==(const __gnu_cxx::__normal_iterator<_IteratorL, _Container>&, const
__gnu_cxx::__normal_iterator<_IteratorR, _Container>&)
    operator==(const __normal_iterator<_IteratorL, _Container>& __lhs,
    ~~~~~~

```

```

/usr/include/c++/7/bits/stl_iterator.h:862:5: note:   template argument deduction/substitution failed:
In file included from /usr/include/c++/7/bits/stl_algobase.h:71:0,
    from /usr/include/c++/7/bits/char_traits.h:39,
    from /usr/include/c++/7/ios:40,
    from /usr/include/c++/7/ostream:38,
    from /usr/include/c++/7/iostream:39,
    from ./headers_getIteratorArbitreParNom.hpp:1,
    from <command-line>:0:

```



```
/usr/include/c++/7/bits/predefined_ops.h:241:17: note:   mismatched types 'const
__gnu_cxx::__normal_iterator<_IteratorL, _Container>' and 'Arbitre* const'
{ return *__it == _M_value; }
~~~~~^~~~~~
```

```
In file included from /usr/include/c++/7/bits/stl_algobase.h:67:0,
      from /usr/include/c++/7/bits/char_traits.h:39,
      from /usr/include/c++/7/ios:40,
      from /usr/include/c++/7/ostream:38,
      from /usr/include/c++/7/iostream:39,
      from ./headers_getIteratorArbitreParNom.hpp:1,
      from <command-line>:0:
```

```
/usr/include/c++/7/bits/stl_iterator.h:869:5: note: candidate: template<class _Iterator, class _Container> bool
__gnu_cxx::operator==(const __gnu_cxx::__normal_iterator<_Iterator, _Container>&, const
__gnu_cxx::__normal_iterator<_Iterator, _Container>&)
operator==(const __normal_iterator<_Iterator, _Container>& __lhs,
~~~~~
```

```
/usr/include/c++/7/bits/stl_iterator.h:869:5: note:   template argument deduction/substitution failed:
```

```
In file included from /usr/include/c++/7/bits/stl_algobase.h:71:0,
      from /usr/include/c++/7/bits/char_traits.h:39,
      from /usr/include/c++/7/ios:40,
      from /usr/include/c++/7/ostream:38,
      from /usr/include/c++/7/iostream:39,
      from ./headers_getIteratorArbitreParNom.hpp:1,
      from <command-line>:0:
```

```
/usr/include/c++/7/bits/predefined_ops.h:241:17: note:   mismatched types 'const
__gnu_cxx::__normal_iterator<_Iterator, _Container>' and 'Arbitre* const'
{ return *__it == _M_value; }
~~~~~^~~~~~
```

```
In file included from /usr/include/x86_64-linux-gnu/c++/7/bits/c++allocator.h:33:0,
      from /usr/include/c++/7/bits/allocator.h:46,
      from /usr/include/c++/7/string:41,
      from /usr/include/c++/7/bits/locale_classes.h:40,
      from /usr/include/c++/7/bits/ios_base.h:41,
      from /usr/include/c++/7/ios:42,
      from /usr/include/c++/7/ostream:38,
      from /usr/include/c++/7/iostream:39,
      from ./headers_getIteratorArbitreParNom.hpp:1,
      from <command-line>:0:
```

```
/usr/include/c++/7/ext/new_allocator.h:155:5: note: candidate: template<class _Tp> bool __gnu_cxx::operator==(
(const __gnu_cxx::new_allocator<_Tp>&, const __gnu_cxx::new_allocator<_Tp>&)
operator==(const new_allocator<_Tp>&, const new_allocator<_Tp>&)
```




```

^~~~~~
/usr/include/c++/7/ext/new_allocator.h:155:5: note:   template argument deduction/substitution failed:
In file included from /usr/include/c++/7/bits/stl_algobase.h:71:0,
                from /usr/include/c++/7/bits/char_traits.h:39,
                from /usr/include/c++/7/ios:40,
                from /usr/include/c++/7/ostream:38,
                from /usr/include/c++/7/iostream:39,
                from ./headers_getIteratorArbitreParNom.hpp:1,
                from <command-line>:0:
/usr/include/c++/7/bits/predefined_ops.h:241:17: note:   mismatched types 'const __gnu_cxx::new_allocator<_Tp>'
and 'Arbitre* const'
    { return *__it == _M_value; }
              ~~~~~^~~~~~

```

Incorrect

Note pour cet envoi : 0,00/1,00.



Écrire la fonction membre **ajouterArbitre** qui ajoute le pointeur d'Arbitre dans les deux attributs conteneurs de la classe **GestionnaireArbitre**. L'ajout doit se faire uniquement si le nom de l'arbitre n'est pas déjà présent. La fonction retourne faux dans le cas où il est déjà présent.

Cette fonction membre doit faire appel à la fonction membre **getArbitreParNom**.

Par exemple:

Test	Résultat
<pre>GestionnaireArbitre gestionArbitre; gestionArbitre.ajouterArbitre(new Arbitre("Martine","principal")); gestionArbitre.ajouterArbitre(new Arbitre("Damien","secondaire")); gestionArbitre.ajouterArbitre(new Arbitre("Raphael","secondaire")); gestionArbitre.ajouterArbitre(new Arbitre("Francois","secondaire")); cout << gestionArbitre;</pre>	<p>le contenu de gestionnaire des arbitres</p> <p>Le conteneur Vector</p> <p>Martine principal</p> <p>Damien secondaire</p> <p>Raphael secondaire</p> <p>Francois secondaire</p> <p>Le conteneur Map</p> <p>secondaire Damien Raphael Francois</p> <p>principal Martine</p>
<pre>GestionnaireArbitre gestionArbitre; gestionArbitre.ajouterArbitre(new Arbitre("albert","principal")); gestionArbitre.ajouterArbitre(new Arbitre("marc","secondaire")); gestionArbitre.ajouterArbitre(new Arbitre("william","secondaire")); gestionArbitre.ajouterArbitre(new Arbitre("charles","principal")); gestionArbitre.ajouterArbitre(new Arbitre("albert","secondaire")); cout << gestionArbitre;</pre>	<p>le contenu de gestionnaire des arbitres</p> <p>Le conteneur Vector</p> <p>albert principal</p> <p>marc secondaire</p> <p>william secondaire</p> <p>charles principal</p> <p>Le conteneur Map</p> <p>secondaire marc william</p> <p>principal albert charles</p>



Test	Résultat
<pre> GestionnaireArbitre gestionArbitre; cout << gestionArbitre.ajouterArbitre(new Arbitre("albert","principal")) << gestionArbitre.ajouterArbitre(new Arbitre("marc","secondaire")) << gestionArbitre.ajouterArbitre(new Arbitre("william","secondaire")) << gestionArbitre.ajouterArbitre(new Arbitre("marc","principal")) << gestionArbitre.ajouterArbitre(new Arbitre("charles","principal")) << gestionArbitre.ajouterArbitre(new Arbitre("albert","secondaire")) << endl; cout << gestionArbitre; </pre>	<pre> 111010 le contenu de gestionnaire des arbitres Le conteneur Vector albert\tprincipal marc\tsecondaire william\tsecondaire charles\tprincipal Le conteneur Map secondaire\tmarc\twilliam\t principal\talbert\tcharles\t </pre>

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```

1  bool GestionnaireArbitre::ajouterArbitre(Arbitre* a)
2  {
3      if (a->getNom() == (getArbitreParNom(a->getNom()))->getNo
4  {
5          return false;
6      }
7      else
8  {
9          arbitres_.push_back(a);
10         return true;
11     }
12 }
```



	Test	Résultat attendu	Résultat obtenu	
✗	<pre>GestionnaireArbitre gestionArbitre; gestionArbitre.ajouterArbitre(new Arbitre("Martine","principal")); gestionArbitre.ajouterArbitre(new Arbitre("Damien","secondaire")); gestionArbitre.ajouterArbitre(new Arbitre("Raphael","secondaire")); gestionArbitre.ajouterArbitre(new Arbitre("Francois","secondaire")); cout << gestionArbitre;</pre>	<pre>le contenu de gestionnaire des arbitres Le conteneur Vector Martine\tprincipal Damien\tsecondaire Raphael\tsecondaire Francois\tsecondaire Le conteneur Map secondaire\tDamien\tRaphael\tFrancois\t principal\tMartine\t</pre>	<pre>***Run error*** Segmentation fault</pre>	✗

Le test a été interrompu à cause d'une erreur.

Montrer les différences

Incorrect

Note pour cet envoi : 0,00/2,00.



La fonction membre **supprimerArbitre**, de la classe **GestionnaireArbitre**, doit supprimer un arbitre des deux conteneurs attributs selon son nom si cet arbitre existe. La fonction retourne vrai si l'arbitre existait et a donc été supprimé.

Compléter le code proprement, en évitant la duplication, sans for/while.

Par exemple:

Test	Résultat
<pre>gestionArbitre.supprimerArbitre("albert"); cout << "suppression" << endl; cout << gestionArbitre;</pre>	<pre>suppression le contenu de gestionnaire des arbitres Le conteneur Vector marc secondaire william secondaire charles principal Le conteneur Map secondaire marc william principal charles</pre>
<pre>gestionArbitre.supprimerArbitre("albert"); gestionArbitre.supprimerArbitre("sophie"); cout << "suppression" << endl; cout << gestionArbitre;</pre>	<pre>suppression le contenu de gestionnaire des arbitres Le conteneur Vector marc secondaire william secondaire charles principal Le conteneur Map secondaire marc william principal charles</pre>
<pre>cout << gestionArbitre.supprimerArbitre("albert") << gestionArbitre.supprimerArbitre("sophie"); cout << endl << "suppression" << endl; cout<< gestionArbitre;</pre>	<pre>10 suppression le contenu de gestionnaire des arbitres Le conteneur Vector marc\tsecondaire william\tsecondaire charles\tprincipal Le conteneur Map secondaire\tmarc\twilliam\t principal\tcharles\t</pre>

Réinitialiser la réponse

```
1 | bool GestionnaireArbitre::supprimerArbitre(string nom) {  
2 |     if(auto it = find(arbitres_.begin(), arbitres_.end(), [nom]  
3 |     {  
4 |         arbitres_.erase(it);  
5 |         return true;  
6 |     }  
7 |     return false;  
8 | }
```

Erreur(s) de syntaxe



```

In file included from /usr/include/c++/7/bits/stl_algobase.h:71:0,
                 from /usr/include/c++/7/bits/char_traits.h:39,
                 from /usr/include/c++/7/ios:40,
                 from /usr/include/c++/7/ostream:38,
                 from /usr/include/c++/7/iostream:39,
                 from ./headers_getIteratorArbitreParNom.hpp:1,
                 from <command-line>:0:
/usr/include/c++/7/bits/predefined_ops.h: In instantiation of 'bool
__gnu_cxx::__ops::_Iter_equals_val<_Value>::operator()(_Iterator) [with _Iterator =
__gnu_cxx::__normal_iterator<Arbitre**, std::vector<Arbitre*> >; _Value = const
GestionnaireArbitre::supprimerArbitre(std::__cxx11::string)::<lambda(Arbitre*)>]':
/usr/include/c++/7/bits/stl_algo.h:120:14:   required from '_RandomAccessIterator
std::__find_if(_RandomAccessIterator, _RandomAccessIterator, _Predicate, std::random_access_iterator_tag) [with
_RandomAccessIterator = __gnu_cxx::__normal_iterator<Arbitre**, std::vector<Arbitre*> >; _Predicate =
__gnu_cxx::__ops::_Iter_equals_val<const GestionnaireArbitre::supprimerArbitre(std::__cxx11::string)::
<lambda(Arbitre*)> >]'
/usr/include/c++/7/bits/stl_algo.h:161:23:   required from '_Iterator std::__find_if(_Iterator, _Iterator,
_Predicate) [with _Iterator = __gnu_cxx::__normal_iterator<Arbitre**, std::vector<Arbitre*> >; _Predicate =
__gnu_cxx::__ops::_Iter_equals_val<const GestionnaireArbitre::supprimerArbitre(std::__cxx11::string)::
<lambda(Arbitre*)> >]'
/usr/include/c++/7/bits/stl_algo.h:3907:28:   required from '_IIter std::find(_IIter, _IIter, const _Tp&) [with
_IIter = __gnu_cxx::__normal_iterator<Arbitre**, std::vector<Arbitre*> >; _Tp =
GestionnaireArbitre::supprimerArbitre(std::__cxx11::string)::<lambda(Arbitre*)>]
__tester__.cpp:2:103:   required from here
/usr/include/c++/7/bits/predefined_ops.h:241:17: error: no match for 'operator==' (operand types are 'Arbitre*'
and 'const GestionnaireArbitre::supprimerArbitre(std::__cxx11::string)::<lambda(Arbitre*)>')
    { return *__it == _M_value; }
          ~~~~~^~~~~~
In file included from /usr/include/c++/7/bits/stl_algobase.h:67:0,
                 from /usr/include/c++/7/bits/char_traits.h:39,
                 from /usr/include/c++/7/ios:40,
                 from /usr/include/c++/7/ostream:38,
                 from /usr/include/c++/7/iostream:39,
                 from ./headers_getIteratorArbitreParNom.hpp:1,
                 from <command-line>:0:
/usr/include/c++/7/bits/stl_iterator.h:862:5: note: candidate: template<class _IteratorL, class _IteratorR,
class _Container> bool __gnu_cxx::operator==(const __gnu_cxx::__normal_iterator<_IteratorL, _Container>&, const
__gnu_cxx::__normal_iterator<_IteratorR, _Container>&)
    operator==(const __normal_iterator<_IteratorL, _Container>& __lhs,
    ~~~~~~
/usr/include/c++/7/bits/stl_iterator.h:862:5: note:   template argument deduction/substitution failed:

```



```

In file included from /usr/include/c++/7/bits/stl_algobase.h:71:0,
                 from /usr/include/c++/7/bits/char_traits.h:39,
                 from /usr/include/c++/7/ios:40,
                 from /usr/include/c++/7/ostream:38,
                 from /usr/include/c++/7/iostream:39,
                 from ./headers_getIteratorArbitreParNom.hpp:1,
                 from <command-line>:0:
/usr/include/c++/7/bits/predefined_ops.h:241:17: note:   mismatched types 'const
__gnu_cxx::__normal_iterator<_IteratorL, _Container>' and 'Arbitre*'
{ return *__it == _M_value; }

```

~~~~~^~~~~~

```

In file included from /usr/include/c++/7/bits/stl_algobase.h:67:0,
                 from /usr/include/c++/7/bits/char_traits.h:39,
                 from /usr/include/c++/7/ios:40,
                 from /usr/include/c++/7/ostream:38,
                 from /usr/include/c++/7/iostream:39,
                 from ./headers_getIteratorArbitreParNom.hpp:1,
                 from <command-line>:0:

```

```

/usr/include/c++/7/bits/stl_iterator.h:869:5: note: candidate: template<class _Iterator, class _Container> bool
__gnu_cxx::operator==(const __gnu_cxx::__normal_iterator<_Iterator, _Container>&, const
__gnu_cxx::__normal_iterator<_Iterator, _Container>&)
operator==(const __normal_iterator<_Iterator, _Container>& __lhs,

```

^~~~~~

```

/usr/include/c++/7/bits/stl_iterator.h:869:5: note:   template argument deduction/substitution failed:

```

```

In file included from /usr/include/c++/7/bits/stl_algobase.h:71:0,
                 from /usr/include/c++/7/bits/char_traits.h:39,
                 from /usr/include/c++/7/ios:40,
                 from /usr/include/c++/7/ostream:38,
                 from /usr/include/c++/7/iostream:39,
                 from ./headers_getIteratorArbitreParNom.hpp:1,
                 from <command-line>:0:

```

```

/usr/include/c++/7/bits/predefined_ops.h:241:17: note:   mismatched types 'const
__gnu_cxx::__normal_iterator<_Iterator, _Container>' and 'Arbitre*'
{ return *__it == _M_value; }

```

~~~~~^~~~~~

```

In file included from /usr/include/x86_64-linux-gnu/c++/7/bits/c++allocator.h:33:0,
                 from /usr/include/c++/7/bits/allocator.h:46,
                 from /usr/include/c++/7/string:41,
                 from /usr/include/c++/7/bits/locale_classes.h:40,
                 from /usr/include/c++/7/bits/ios_base.h:41,
                 from /usr/include/c++/7/ios:42,

```




```

        from /usr/include/c++/7/ostream:38,
        from /usr/include/c++/7/iostream:39,
        from ./headers_getIteratorArbitreParNom.hpp:1,
        from <command-line>:0:
/usr/include/c++/7/ext/new_allocator.h:155:5: note: candidate: template<class _Tp> bool __gnu_cxx::operator==(
(const __gnu_cxx::new_allocator<_Tp>&, const __gnu_cxx::new_allocator<_Tp>&)
    operator==(const new_allocator<_Tp>&, const new_allocator<_Tp>&)
        ^~~~~~
/usr/include/c++/7/ext/new_allocator.h:155:5: note:   template argument deduction/substitution failed:
In file included from /usr/include/c++/7/bits/stl_algobase.h:71:0,
        from /usr/include/c++/7/bits/char_traits.h:39,
        from /usr/include/c++/7/ios:40,
        from /usr/include/c++/7/ostream:38,
        from /usr/include/c++/7/iostream:39,
        from ./headers_getIteratorArbitreParNom.hpp:1,
        from <command-line>:0:
/usr/include/c++/7/bits/predefined_ops.h:241:17: note:   mismatched types 'const __gnu_cxx::new_allocator<_Tp>'
and 'Arbitre*'
    { return *__it == _M_value; }
        ~~~~~^~~~~~

```

Incorrect

Note pour cet envoi : 0,00/2,00.



Question **16**

Incorrect

Note de 0,00
sur 2,00

Compléter proprement la surcharge de l'opérateur <<, il y a le caractère "\t" entre les informations, mais pas à la fin des lignes. Préférer les boucles sur intervalles (celles avec :) aux anciens **for** (avec ; ;). Note: le compilateur supporte bien C++17.

Par exemple:

| Test | Résultat |
|---|--|
| <pre>GestionnaireArbitre gestionArbitre; gestionArbitre.ajouterArbitre(new Arbitre("albert","principal")); gestionArbitre.ajouterArbitre(new Arbitre("marc","secondaire")); gestionArbitre.ajouterArbitre(new Arbitre("william","secondaire")); gestionArbitre.ajouterArbitre(new Arbitre("charles","principal")); cout<< gestionArbitre;</pre> | <pre>le contenu de gestionnaire des arbitres Le conteneur Vector albert\tprincipal marc\tsecondaire william\tsecondaire charles\tprincipal Le conteneur Map secondaire\tmarc\twilliam principal\talbert\tcharles</pre> |

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1 | ostream& operator<< (ostream& sortie, const GestionnaireArbit
2 | {
3 |     sortie << "le contenu de gestionnaire des arbitres" << en
4 |     sortie << "Le conteneur Vector" << endl;
5 |     for (auto it : range(arbitres_))
6 |     {
7 |         sortie << arbitres_[it];
8 |     }
9 |     sortie << "Le conteneur Map" << endl;
10 |    for(const auto& [key, value] : filtreArbitreRole_)
11 |    {
12 |        sortie << "Rôle de l'arbitre: " << key << ", Arbitres
13 |        for (auto it : range(value))
14 |        {
15 |            sortie << value[it];
16 |        }
17 |    }
18 |    sortie << endl;
19 |    return sortie;
20 | }
```



Erreur(s) de syntaxe

```
__tester__.cpp: In function 'std::ostream& operator<<(std::ostream&, const GestionnaireArbitre&)':
__tester__.cpp:5:26: error: 'arbitres_' was not declared in this scope
    for (auto it : range(arbitres_))
                        ^~~~~~
__tester__.cpp:5:26: note: suggested alternative: 'Arbitre'
    for (auto it : range(arbitres_))
                        ^~~~~~
                        Arbitre
__tester__.cpp:5:20: error: 'range' was not declared in this scope
    for (auto it : range(arbitres_))
                    ^~~~~
__tester__.cpp:5:20: note: suggested alternative: 'rand'
    for (auto it : range(arbitres_))
                    ^~~~~
                    rand
__tester__.cpp:10:36: error: 'filtreArbitreRole_' was not declared in this scope
    for(const auto& [key, value] : filtreArbitreRole_)
                                   ^~~~~~~~~~~~~~~~~~~~~
__tester__.cpp:13:24: error: 'range' was not declared in this scope
    for (auto it : range(value))
                      ^~~~~
__tester__.cpp:13:24: note: suggested alternative: 'rand'
    for (auto it : range(value))
                      ^~~~~
                      rand
```

Incorrect

Note pour cet envoi : 0,00/2,00.



Question **17**

Terminé

Non noté

Sur mon honneur, je (écrire votre nom) , affirme que cet examen par moi-même, sans communication avec personne (autre que les enseignants indiqués en première page en cas de problème), et selon les directives identifiées dans cet énoncé d'examen.

Question **18**

Non répondue

Non noté

Si nécessaire, inscrivez vos suppositions ici, en précisant pour chaque supposition le numéro de la question concernée.



