

Aucun document n'est autorisé à part la fiche résumé de C++, où vous pouviez consigner des notes manuscrites personnelles au verso. Tous les exercices sont indépendants. Même si l'on ne sait pas répondre à une question, on peut utiliser la réponse dans la suite de l'exercice. Une grande importance sera accordée à la qualité de la rédaction (lisibilité, indentation,...).

Le barème est indicatif et pourra changer à la correction.

Durée : 2h00.

► **Exercice 1. (Compilation séparée)** – sur 5 points –

Une équipe de développeurs travaille sur des systèmes de calculs. Ils doivent développer une application de calculatrice capable de manipuler des fractions dont les numérateurs et dénominateurs peuvent être très grands ; Pour ceci, ils vont écrire les classes suivantes :

- **Entier** : une classe pour les entiers en précision arbitraire ;
- **Rational** : une classe pour les fractions dont les numérateurs et dénominateurs sont des **Entier** ;
- **Memoire** : une classe pour gérer la mémoire de la calculatrice ; cette classe contient un tableau de **Rational** ;

Chacune de ces classes devra être compilée dans un fichier à part. Pour chacune de ces classes on a écrit la déclaration dans un fichier portant le nom de la classe et l'extension `.hpp` et les définitions dans un fichier portant le nom de la classe et l'extension `.cpp`. Par exemple **Entier** est dans **Entier.hpp** et **Entier.cpp**

En outre, ils ont écrit le programme principal (`main`) dans le fichier `calc.cpp`.

1. Pour chacun des 7 fichiers, on demande d'écrire les directives d'inclusion (`include`) des fichiers qui ont été développés par l'équipe. On ne demande pas d'écrire les inclusions de la bibliothèque standard (par exemple `iostream`, `vector`).
2. Quelle directive d'inclusion doit-on mettre pour que l'on puisse manipuler un tableau dans la classe **Mémoire** ?
3. Quelle est la commande qui permet de compiler la classe **Entier**. Quel est le nom du fichier produit ?
4. Quelle est la commande qui permet de produire l'exécutable pour la calculatrice ?
5. Écrire les deux lignes de `Makefile` qui concernent la compilation de la classe **Memoire**.

Problème : Réservation de chambres d'étudiants

Le but de ce problème est de participer à la construction d'un système de réservation de chambre dans une résidence pour étudiants. La résidence dispose d'un nombre fixé `nb_chambres` de chambres.

► Exercice 2. (Classe pour représenter les d'étudiants) – sur 4 points –

Un étudiant est modélisé par la classe `Etudiant`. Quand on crée un étudiant, on connaît son nom (une chaîne de caractères) et son numéro d'étudiant (un entier). Ils ne seront jamais modifiés. En revanche, il n'a pas encore de chambre affectée. On pourra ensuite lui affecter une chambre (le numéro de chambre est un entier).

1. Écrire la déclaration du constructeur de la classe `Etudiant`.
2. Écrire les déclarations des méthodes qui permettent d'accéder aux nom, numéro et chambre d'un étudiant, ainsi que celles des méthodes qui permettent de les modifier (uniquement pour les attributs où cela est utile).
3. Dans quel fichier les déclarations des constructeurs et des méthodes d'accès et de modification doivent-elles figurer ? A quel endroit du programme très précisément ?
4. Quand un étudiant n'a pas de chambre, on lui donne le numéro de chambre 0. Déclarer une méthode `est_loge` qui renvoie si l'étudiant à une chambre ou non.
5. Définir la méthode `est_loge`.
6. À quel endroit et dans quel fichier doit-on placer cette définition ?

► Exercice 3. (Représentation des Chambres) – sur 1 points –

On s'intéresse maintenant à la structure pour représenter les chambres. La résidence comporte 100 chambres réparties sur deux étages. Les chambres de l'étage 1 sont numérotées de 100 à 149. Les chambres de l'étage 2 sont numérotées de 200 à 249. Pour stocker les informations des chambres dans des tableaux, on a besoin d'associer à chaque chambre un *indice* entre 0 et 99. Ainsi les chambres de l'étage 1 correspondent aux indices de 0 à 49 et les chambres de l'étage 2 aux indices de 50 à 99. Par exemple, la chambre 127 a pour indice 27 et la chambre 233 a pour indice 83.

7. Écrire une fonction `indice_chambre` qui renvoie l'indice d'une chambre connaissant son numéro. Si le numéro de la chambre n'est pas valide, on signalera une exception.

Dans la suite on suppose écrite une fonction similaire `numero_chambre` qui renvoie le numéro d'une chambre connaissant son indice.

Voici la partie publique de la déclaration de la classe `Residence` :

```
// Constructeur
Residence();
// Renvoie true si la chambre du numéro donné est libre
bool est_libre(int numero) const;
// Réserve une chambre pour l'étudiant
void reserve(Etudiant &etu);
```

```
// Libère la chambre de l'étudiant
void libere(Etudiant &etu);
```

► **Exercice 4. (Implémentation simple à un seul tableau)** – sur 5.75 points –

Dans une première partie on représente les chambres par un attribut privé de la classe `Residence`, nommé `chambres`, qui est un tableau d'entiers. Dans la case `i` du tableau on notera le numéro de l'étudiant qui occupe la chambre, ou 0 si la chambre n'est pas occupée.

8. Écrire la déclaration de l'attribut `chambres`. Dire très précisément où cette déclaration doit être écrite.
9. Écrire la définition du constructeur de la classe `Residence`.
10. Écrire la définition de la méthode `est_libre`.
11. Écrire la définition de la méthode `reserve`. L'étudiant ne doit pas avoir déjà une chambre, sinon on lèvera une exception. D'autre part, s'il n'y a pas de chambre libre, on lèvera également une exception. L'étudiant est passé par référence à la méthode car on doit modifier l'étudiant pour lui donner sa chambre.
12. Écrire la définition de la méthode `libere`. Si l'étudiant n'a pas de chambre, on lèvera une exception. L'étudiant est passé par référence à la méthode car on doit modifier l'étudiant pour lui enlever sa chambre.

► **Exercice 5. (Implémentation avec un tableau et un vecteur)** – sur 4.25 points –

L'université de Marie-sa-Clé souhaite utiliser cette application. Malheureusement, elle ne possède pas 100 chambres mais plutôt 4000. Si les développeurs ont pu facilement adapter les fonctions `indice_chambre` et `numero_chambre`, l'implémentation précédente se révèle trop lente à l'usage. Les développeurs décident de faire une deuxième implémentation qui utilise en plus du tableau `chambres`, un vecteur `libres` qui contient les indices des chambres libres, dans un ordre quelconque. Quand on libère une case on rajoute son indice à la fin du vecteur. Quand on réserve une case, on réserve celle dont l'indice est à la fin du vecteur. On n'a donc jamais besoin de décaler les éléments du vecteur.

Avec cette nouvelle solution, on demande de ré-écrire les définitions ci-dessous. Bien sûr, l'interface de ces méthodes ne doit pas changer. De plus, pour que le programme aille le plus vite possible, on s'efforcera de ne pas utiliser de boucle.

13. la définition du constructeur de la classe `Residence`.
14. la définition de la méthode `reserve`.
15. la définition de la méthode `libere`.