Examen AP4A – Automne 2023

Durée 1h30

Préambule :

- Les exercices 1 et 2 sont indépendants et doivent être rédigés sur des feuilles séparées.
- Pas de document, ni de calculatrice autorisée. Dictionnaire papier autorisé.

Exercice 1 (10 points)

Dans les calls center, les appels entrants sont mis en attente pour être répondus par un opérateur téléphonique. De la même manière, les processus devant être exécutés par un ordinateur sont mis en attente pour être traités. Une file d'attente est mis en œuvre à chaque fois pour les traiter dans l'ordre d'arrivée.

En informatique, une file d'attente est une structure de données abstraite qui contient une collection d'éléments. La file d'attente implémente le mécanisme FIFO, c'est-à-dire que l'élément inséré en premier est également supprimé en premier. En d'autres termes, l'élément le moins récemment ajouté est supprimé en premier dans une file d'attente.

Dans cette partie, il s'agit de donner une implémentation de la classe FILE.

La classe FILE devra réutiliser le code d'une classe Liste, une liste chaînée. Ainsi la classe File sera une classe Liste spéciale avec des opérations limitées :

Enfiler() : ajoute un nouvel élément à la fin de la file d'attente.
Défiler() : retire un élément du début de la file et retourne sa valeur.
Premier() : retourne la valeur de l'élément au début de la file.
Taille() : retourne le nombre d'éléments dans la file.
estVide() : retourne 1 si la file est vide, sinon retourne 0.

Donnez les implémentations des classes Liste et File génériques.

Exercice 2 (10 points)

On souhaite développer une application de type questionnaire permettant de créer des tests de connaissances. Le questionnaire se présente sous la forme de couples (« question », « réponse »), la « question » étant une chaîne de caractères constituée de la question à poser et des différentes réponses possibles numérotées à partir de 1, et la « réponse » étant un entier (≥ 1) correspondant à la réponse correcte. Exemple de question :

```
Is it possible to override operators in C++ ?
1. Yes
2. No
```

On définit tout d'abord la classe « question_answer » qui associe une question à la bonne réponse correspondante :

```
class question_answer {
    private :
        std::string question ; // question to ask
```

```
        int answer ; // the right answer

    public :
        question_answer():question(""),answer(0){}
};
```

1. Compléter la classe « question_answer » pour que le code qui suit puisse fonctionner correctement. Le 2ème paramètre du constructeur utilisé ci-dessous correspond à la bonne réponse à la question. L'affichage de l'objet « question_answer » affichera d'abord la question puis la réponse attendue sur la ligne suivante.

```
    question_answer qa("Is it possible to override operators in C++ ?\n1. Yes\n2. No", 1) ;
    std::cout << qa << endl ;
```

2. Définir la méthode « bool question_answer::ask() » qui permet d'afficher la question à l'écran, de demander la réponse à l'utilisateur et qui renvoie « true » or « false » suivant que la réponse est correcte ou pas.
3. Faut-il définir la forme canonique de Coplien pour cette classe ? Si oui, donner les méthodes correspondantes.
4. On souhaite maintenant créer la classe « questionnaire » qui contiendra un tableau dynamique de plusieurs questions/réponses à partir de l'embryon de classe qui suit. Compléter les attributs de cette classe et donner son destructeur.

```
class questionnaire {
    private:
       question_answer* tab;
       ...
} ;
```

5. Surcharger l'opérateur « << » afin de pouvoir ajouter des questions/réponses au questionnaire de la manière suivante :

```
    questionnaire quest ;
    quest << question_answer("New question ?\n1. Yes\n2. No", 1) ;
```

6. Surcharger l'opérateur « << » afin de pouvoir afficher l'ensemble des questions/réponses d'un questionnaire

```
    questionnaire quest ;
    ...
    std::cout << quest ;
```

7. Ecrire la méthode « float questionnaire::ask() » qui pose toutes les questions à l'utilisateur et retourne le score final à la fin sous forme de pourcentage de bonnes réponses obtenues.

```
    questionnaire quest ;
    ...
    std::cout << "Percentage of correct answers=" << quest.ask() ;
```

8. On souhaite maintenant définir une nouvelle classe de questionnaire dont les réponses sont des chaînes de caractères de type « string ». Proposer un mécanisme C++ qui permette d'aboutir aisément à ce résultat tout en gardant également la possibilité d'avoir une classe de questionnaire dont les réponses sont des entiers.