IFT2015-H26 - Quiz #1 - 15 janvier 2026

Nom: Prénom:

Matricule:

Directives:

- Répondez directement sur cette feuille
- Aucune documentation ni appareil électronique permis.

(6 points) Question 1 : Expliquez ce qu'est un type abstrait de données (ADT) et ce qu'est un invariant dans le contexte des structures de données. Expliquez ensuite, à l'aide d'un exemple de votre choix, comment un invariant permet d'accélérer certaines opérations.

Un type abstrait de données (ADT) définit un ensemble d'opérations et leur comportement, indépendamment de la manière dont elles sont implémentées. Une structure de données concrète est une implémentation d'un ADT.

Un invariant est une propriété qui est toujours vraie pour une structure valide. Il impose une organisation particulière des données.

Par exemple, dans une liste triée, l'invariant est que les éléments sont toujours en ordre. Cet invariant permet d'accélérer la recherche (par exemple par recherche dichotomique), au prix d'un coût plus élevé lors des insertions.

(9 points) Question 2 : Considérez le code suivant :

```
List<Integer> list = new SinglyLinkedList<>();
for( int i = 0; i < n; i++ ) {
    list.add( i );
}
int sum = 0;
for( int i = 0; i < n; i++ ) {
    sum += list.get( i );
}
```

a) Donnez la complexité asymptotique du deuxième **`for`**. Justifiez votre réponse.

Complexité asymptotique : $O(n^2)$

add(i) ajoute en *O(1)* un nouvel élément à la fin de la liste => *O(n)* pour le premier **`for`**.

`get( i )` parcourt la liste pour arriver au ième élément => *O(n)*. Pour le 2ème **`for`**, on applique `get( i )` pour *n* éléments => $O(n^2)$.

$O(n) + O(n^2) = O(n^2)$

b) Proposez une modification simple du code permettant d'améliorer la complexité, sans changer le résultat final.

On parcourt la liste avec l'itérateur :

```
for( int x : list ) {
    sum += x;
}
```

Cette version parcourt la liste une seule fois et a une complexité de *O(n)*.

OU

On utilise un `ArrayList` au lieu d'une `SinglyLinkedList`. Dans l'implémentation de `ArrayList`, `get( i )` est en *O(1)*, donc $n * O(1) = O(n)$.

Barème

Question 1 (6 points)

- ADT correctement défini : 2 pts
- Invariant correctement défini : 2 pts
- Exemple pertinent : 2 pts

Question 2 (9 points)

- Analyse correcte : 4 pts
- Justification claire : 3 pts
- Amélioration correcte : 2 pts