Correction du Quick 1

Enoncé

On cherche ici à étudier de manière plus précise l'algorithme suivant qui opère des déplacements d'entiers dans une séquence sous forme de tableau avec longueur explicite.
Pour chaque élément de la séquence dans l'ordre : si l'élément est pair, le déplacer en fin de séquence.

Voici ci-dessous un exemple de début d'exécution. (L'élément souligné est le prochain qui sera testé.)

Etape 1 : <**1**, 8, 3, 4, 6, 2, 7>
Etape 2 : <1, **8**, 3, 4, 6, 2, 7>
Etape 3 : <1, **3**, 4, 6, 2, 7, 8>
Etape 4 : <1, 3, **4**, 6, 2, 7, 8>
Etape 5 : <1, 3, **6**, 2, 7, 8, 4>
.
.
.

Question 1 :

Question de cours : donnez les définitions des structures de données qui seront nécessaires

La structure de donnée nécessaire est une **séquence** car dans cette énoncé les éléments vont changer de place. L'ordre des éléments dans la structure de donnée est donc important. Une structure **d'ensemble** ne peut donc pas être utilisée.

L'énoncé donne comme indication d'utilisé un structure de donnée bas-niveau sous forme de tableau.
La séquence est donc défini par :

type Séquence : {
    tab : tableau de LMAX éléments (avec LMAX un entier très grand),
    longueur : entier ≥ 0
}

Question 2 :

Terminez le déroulement de l'exemple précédent, mais cette fois en **détaillant la représentation dans la mémoire** : explicitez à chaque étape le tableau, son contenu, les valeurs des variables dont vous aurez Besoin (par exemple les indices des éléments), et les déplacements de valeurs effectués.

Etape 6 :

i = 2

1	3	~~6~~	2	7	8	4

$x \leftarrow 6$

$j = 5$

i = 2

1	3	2	7	8	4	6

Etape 7 :

i = 2

1	3	~~2~~	7	8	4	6

$x \leftarrow 2$

$j = 6$

i = 2

1	3	7	8	4	6	2

Question 2 :

Terminez le déroulement de l'exemple précédent, mais cette fois en **détaillant la représentation dans la mémoire** : explicitez à chaque étape le tableau, son contenu, les valeurs des variables dont vous aurez Besoin (par exemple les indices des éléments), et les déplacements de valeurs effectués.

Etape 8 :	j est égale à la taille de la séquence donc on s'arrête	$j = 7$

Question 3 : Haut-niveau

Donnez l'algorithme en pseudo-code au dos de cette feuille. **Attention** : il est conseillé de donner l'algorithme principal en haut-niveau et de détailler les opérations bas-niveau dans des fonctions séparées.

Question 3 : Haut-niveau

On cherche ici à étudier de manière plus précise l'algorithme suivant qui opère des déplacements d'entiers dans une séquence sous forme de tableau avec longueur explicite.
Pour chaque élément de la séquence dans l'ordre : si l'élément est pair, le déplacer en fin de séquence.

Une première idée :

Pour chaque élément de la séquence :
    si l'élément est pair :
        le déplacer en fin de séquence

Déplacer_pair(S : Séquence d'entier)
    Pour chaque élément e ∈ S **faire**
        si e est pair **alors**
            déplacer(S, e)

Juste une réindentation de l'énoncer

Une écriture un peu plus clair mais qui ne permet pas encore de décrire toute les subtilités de l'algorithme.

Question 3 : Haut-niveau

On cherche ici à étudier de manière plus précise l'algorithme suivant qui opère des déplacements d'entiers dans une séquence sous forme de tableau avec longueur explicite.
Pour chaque élément de la séquence dans l'ordre : si l'élément est pair, le déplacer en fin de séquence.

```
Déplacer_pair(S : Séquence d'entier)
    Pour chaque élément e ∈ S faire
        si e est pair alors
            déplacer(S, e)
```

Une écriture un peu plus clair mais qui ne permet pas encore de décrire toute les subtilités de l'algorithme.

```
Déplacer_pair(S : Séquence d'entier)
    i ⟵ 0
    Pour j allant de 0 à taille(S) faire
        x ⟵ obtenir(S, i)
        si x modulo 2 = 0 alors
            déplacer(S, i)
            ajouter_fin(S, x)
        sinon
            i ⟵ i + 1
```

Une écriture clair qui montre les subtilités de l'algorithme en utilisant les primitives des séquences pour rester haut-niveau.

Question 3 : Bas-niveau

Haut-niveau :

```
Déplacer_pair(S : Séquence d'entier)
    i ⟵ 0
    Pour j allant de 0 à taille(S) faire
        x ⟵ obtenir(S, i)
        si x modulo 2 = 0 alors
            déplacer(S, i)
            ajouter_fin(S, x)
        sinon
            i ⟵ i + 1
```

Bas-niveau :

```
obtenir(S : Séquence d'entier, i : entier) : entier
    retourner S.tab[i]


déplacer(S : Séquence d'entier, i : entier)
    Pour j allant de i à S.longueur-2 faire
        S.tab[i] ⟵ S.tab[i+1]
    S.longueur ⟵ S.longueur-1


ajouter_fin(S : Séquence d'entier, x : entier)
    S.tab[S.longueur] ⟵ x
    S.longueur ⟵ S.longueur+1
```

Question 4 : Complexité

Haut-niveau :

```
Déplacer_pair(S : Séquence d'entier)
    i ⟵ 0
    Pour j allant de 0 à taille(S) faire
        x ⟵ obtenir(S, i)  O(1)
        si x modulo 2 = 0 alors
            déplacer(S, i)  O(n)
            ajouter_fin(S, x)  O(1)
        sinon
            i ⟵ i + 1
```

$O(n^2)$

Bas-niveau :

```
obtenir(S : Séquence d'entier, i : entier) : entier   O(1)
    retourner S.tab[i]
```

```
déplacer(S : Séquence d'entier, i : entier)
    Pour j allant de i à S.longueur-2 faire
        S.tab[i] ⟵ S.tab[i+1]
    S.longueur ⟵ S.longueur-1
```

$O(n)$

```
ajouter_fin(S : Séquence d'entier, x : entier)   O(1)
    S.tab[S.longueur] ⟵ x
    S.longueur ⟵ S.longueur+1
```

La complexité finale de l'algorithme est de $O(n^2)$.

Question 5 : Optimisation (version textuelle)

Problématique:
L'algorithme précédent possède une complexité en $O(n^2)$ à cause du décalage nécessaire pour déplacer un élément à la fin de la séquence. Il faut donc trouver un moyen de supprimer ce décalage

Structure de donnée :
Nous choisissons ici d'utiliser une structure de donnée haut-niveau sous forme de séquence avec une implémentation bas-niveau à base de liste chainée tels que vue dans le cours en ajoutant un champ queue contenant la référence vers la dernière cellule.

Algorithme :
L'idée de l'algorithme est de parcourir la liste chainée et de tester à chaque itération si la valeur de la cellule est pair. Si elle est pair nous changeons le chaînage en utilisant la référence courante et précédente ainsi :

- La cellule en queue pointe vers la cellule courante
- La cellule courante pointe vers Nil
- La cellule précédente pointe vers la cellule suivant la cellule courante
- La référence courante pointe vers la cellule suivant la cellule courante

Question 5 : Optimisation (version graphique)