TD/TP sur les séquences implémentées par des listes chaînées.

1 TD : Rappel d'algorithmes vus en cours et implémentations.

En cours, vous avez vu les premiers algorithmes sur les listes chaînées. Dans un premier temps, il vous est demandé de rappeler ces algorithmes en langage algo (vous pouvez ajouter des schémas et des analyses de complexité). Dans un second temps, il vous est demandé de traduire (implémenter) ces algorithmes dans un langage de programmation particulier (en C pour les parcours INF et MIN, en Python pour le parcours MAT) pour une implémentation utilisant des listes chaînées.

La structure de données conseillée, en langage C, pour les listes chaînées d'entiers est la suivante :

```c
struct cellule {
    int valeur;
    struct cellule *suivant;
};
typedef struct cellule cellule_t;
```

```c
struct liste {
    cellule_t *tete;
};
typedef struct liste liste_t;
```

La structure de données conseillée en Python est la suivante :

```python
class Cellule:
    def __init__(self):
        self.valeur = None
        self.suivant = None
```

```python
class Liste:
    def __init__(self):
        self.tete = None
```

Exercice 1.1. Ajout en tête [Schéma de base]. Rappeler l'algorithme pour ajouter une valeur en tête d'une séquence puis fournir une implémentation de cet algorithme pour les listes chaînées.

Profil de la fonction (en C) :

```c
void ajout_en_tete(liste_t *l, int val);
```

Exercice 1.2. Ajout en queue [Schéma de base]. Rappeler l'algorithme pour ajouter une valeur en fin d'une séquence sous forme de liste chaînée puis fournir une implémentation de cet algorithme pour les listes chaînées.

Profil de la fonction (en C) :

```c
void ajout_en_queue(liste_t *l, int val);
```

2 TD : D'autres algorithmes et implémentations.

Ici les algorithmes ne sont pas donnés, c'est la première étape des exercices, la seconde reste l'implémentation. Les deux étapes peuvent être faites en même temps.

Exercice 2.1. Parcours [Schéma de base]. Donner un algorithme et son implémentation pour faire la somme des valeurs des cellules d'une séquence d'entiers implémentée par une liste chaînée.

Profil de la fonction (en C) :

```c
int somme(liste_t *l);
```

Variante : Calculer la moyenne des valeurs.

Pour aller plus loin, un cas concret : La séquence représente les altitudes lors d'un trail en montagne, calculer le dénivelé positif global.

Exercice 2.2. Recherche de référence. Donner un algorithme et son implémentation pour déterminer la référence de la première cellule qui comporte une valeur nulle (0) dans une séquence sous forme de liste chaînée. Dans le cas où la séquence ne comporte pas de valeur nulle (0), retourner la référence nulle (`NULL` en C ou `None` en Python).

Profil de la fonction (en C) :

```
cellule_t * reference_zero(liste_t *l);
```

Indication : la référence de la première cellule est donnée par la tête, celle des cellules suivantes est donnée par le chaînage **suivant** de la cellule précédente (n.b. : il est parfois utile de déterminer la référence recherchée à partir de son prédécesseur).

Exercice 2.3. Insertion avant. Donner un algorithme et son implémentation pour insérer une valeur nulle (zéro) avant la première valeur négative et au plus tard en dernière valeur.

Profil de la fonction (en C) :

```
void insertion_zero(liste_t *l);
```

3 TP : Implémentations et test.

Le TP a lieu sur Caseine, section "Semaine de Transition - Listes chaînées". Vous y trouverez des exercices immédiatement issus du TD et quelques exercices supplémentaires.

Exercice 3.1. Ajout en tête [Schéma de base]. Implémenter et tester l'ajout d'une valeur en tête d'une séquence implémentée sous forme de liste chaînée. La saisie et l'affichage de la séquence sont fournis.

Exercice 3.2. Ajout en queue [Schéma de base]. Implémenter et tester l'ajout d'une valeur en fin d'une séquence implémentée sous forme de liste chaînée. La saisie et l'affichage de la séquence sont fournis.

Exercice 3.3. Insertion dans une liste triée [Schéma de base]. Implémenter et tester l'ajout d'une valeur dans une séquence triée croissante. La saisie et l'affichage de la séquence (triée) sont fournis.

Exercice 3.4. Suppression des doublons consécutifs. Implémenter et tester la suppression des doublons consécutifs d'une séquence implémentée sous forme de liste chaînée. La saisie et l'affichage de la séquence sont fournis.

Exercice 3.5. Inversion d'une liste. Implémenter et tester l'inversion d'une séquence implémentée sous forme de liste chaînée. La saisie et l'affichage de la séquence sont fournis.

Exercice 3.6. Tri d'une liste. Implémenter et tester le tri d'une séquence d'entiers implémentée sous forme de liste chaînée. La saisie et l'affichage de la séquence sont fournis.