Révisions

I. Échauffement

Question 1 Écrire des fonctions qui :
- ☞ détermine si un élément est présent dans une liste;
- ☞ renvoie le dernier élément d'une liste non vide;
- ☞ concatène deux listes (sans utiliser `@`);
- ☞ calcule le miroir d'une liste.

II. Listes triées

On considère dans cette partie que l'on a des listes triées sans répétition.

Question 2 En considérant que les listes représentent des ensembles, écrire des fonctions qui calculent :
- ☞ l'intersection de deux ensembles;
- ☞ l'union de deux ensembles;
- ☞ la différence symétrique de deux ensembles.

Question 3 Écrire une fonction qui ajoute un élément dans une liste triée sans répétition.

Question 4 En déduire une implémentation du tri par insertion.

III. Élément majoritaire dans un tableau

Étant donné un tableau `t` de taille n, on dit qu'un élément e est **majoritaire** dans `t` s'il y apparaît strictement plus que $\frac{n}{2}$ fois. Bien sûr, un tel élément majoritaire n'existe pas toujours et s'il existe, il est unique.

Nous allons étudier plusieurs algorithmes permettant de déterminer si un tableau contient un élément majoritaire et, si oui, lequel.

1. Approche naïve

Commençons, comme souvent, par une approche simple mais pas forcément efficace.

Question 5 Écrire une fonction `occurences` telle que `occurence t x` renvoie le nombre d'occurences de `x` dans le tableau `t`.

Question 6 En déduire une fonction

```
em1 : 'a array -> 'a option
```

qui, étant donné un tableau en argument, indique s'il contient un élément majoritaire et, si oui, indique lequel.

Question 7 Indiquer sa complexité.

2. Diviser pour régner

Nous allons maintenant adapter l'algorithme précédent en appliquant la méthode « diviser pour régner. »

Question 8 Déterminer la relation entre l'éventuel élément majoritaire d'un tableau `t` et ceux des première et deuxième moitiés de `t`.

Question 9 En déduire une fonction récursif `em2` qui détermine l'existence et la valeur éventuelle de l'élément majoritaire de `t`.

Question 10 Indiquer sa complexité.

3. L'algorithme de Boyer-Moore pour déterminer l'élément majoritaire

Nous supposons maintenant que l'on **sait** que le tableau `t` contient un élément majoritaire.

L'algorithme de Boyer et Moore peut se décrire ainsi. On a deux variables :

- `candidat` dont la valeur est le candidat actuel pour l'élément majoritaire ;
- `compte` qui est entière, initialisée à 0.

On parcourt le tableau de gauche à droite, et pour chaque valeur `v` du parcours,

- si $\texttt{compte} = 0$, on affecte `candidat` à la valeur `v` et on incrémente `compte` ;
- sinon, si $\texttt{candidat} = \texttt{v}$, on incrémente `compte` et on le décrémente sinon.

Question 11 Prouver la correction de l'algorithme, c'est-à-dire que si le tableau contient un élément majoritaire, c'est bien lui qui est retourné. On pourra considérer la valeur de c définie par :

$$c = \begin{cases} \texttt{compte} & \text{si } \texttt{candidat} = e\,; \\ -\texttt{compte} & \text{sinon.} \end{cases}$$

Question 12 En déduire une fonction `em3` de **complexité linéaire** qui détermine l'existence et la valeur éventuelle de l'élément majoritaire de `t`.