Important. Une grande importance sera apportée à la clarté, la lisibilité des raisonnements et des programmes. Dans ce sujet, on rédigera les programmes dans le langage OCaml.

Remarque. On pourra toujours librement utiliser une fonction demandée à une question précédente, même si cette question n'a pas été traitée.

Exercice 1 (Fonctions sur les listes) : Dans cet exercice, on veut implémenter nous même certaines fonctions sur les listes qui existent dans la librairie `List` d'OCaml.

1. Écrire une fonction `iter : ('a -> unit) -> 'a list -> unit` qui prend en entrée une fonction `f` de type `'a -> unit` et une liste `l` d'éléments de type `'a`, et appelle la fonction `f` successivement sur les éléments de `l` (dans l'ordre).
   Autrement dit, `iter f [a1 ; a2 ; ... an]` doit être équivalent à : `f a1 ; f a2 ; ... ; f an`.
2. Écrire une fonction `filter : ('a -> bool) -> 'a list -> 'a list` qui prend en entrée une fonction `f` de type `'a -> bool` (`f` est un prédicat), et une liste `l` d'éléments de type `'a`, et qui renvoie la liste des éléments de `l` qui vérifient le prédicat `f` (dans l'ordre).
3. Écrire une fonction `append : 'a list -> 'a list -> 'a list` telle que `append l1 l2` renvoie une nouvelle liste issue de la concaténation de `l1` et `l2` (l'utilisation de l'opérateur `@` est interdit).
4. Écrire une fonction `flatten : 'a list list -> 'a list` qui prend une liste de listes en entrée et renvoie la concaténation de toutes ces listes (dans l'ordre).

Exercice 2 (Fonction de Hofstadter) : On considère la fonction de Hofstadter suivante, dont on souhaite étudier la terminaison :

```
let rec h n = match n with
  | 0 -> 0
  | _ -> n - h(h(n-1))
;;
```

1. Vérifier que `h` termine sur 0, 1, 2, 3, 4 et donner les valeurs renvoyées.
2. Démontrer que `h` termine sur tout entier naturel.
   Indication : on pourra montrer que $\forall n \in \mathbb{N}$, `h` termine et de plus renvoie une valeur appartenant à un intervalle dépendant de n à déterminer.

Problème : Élection par majorité

On souhaite analyser les résultats de sondages concernant une élection. Les candidats à l'élection sont numérotés de 0 à $k-1$. Les résultats de chaque sondage sont stockés dans un tableau : si le sondage a recueilli N réponses, le tableau comporte N cases, une pour chaque réponse : la i-ème case du tableau contient le numéro du candidat proposé par la i-ème personne sondée. On a ainsi un tableau T de longueur N qui contient des entiers de $[\![0, k-1]\!]$.

Le sondage donne le candidat numéroté $a \in [\![0, k-1]\!]$ élu si le nombre a est dans strictement plus de $N/2$ cases de T. Par exemple, un sondage correspondant au tableau $[2, 4, 5, 0, 4, 4, 4]$ donne le candidat 4 élu. Mais un tableau ne donne pas toujours un élu, par exemple le tableau $[1, 2, 3, 4, 6, 2, 3, 3]$.

On étudie différentes manières de calculer la valeur élue par un tableau, si elle existe.

Exercice 3 (Méthode naïve) :

1. (a) Écrire une fonction `nb` prenant un tableau `t` de type `int array` et un entier `a`, et renvoyant le nombre de fois qu'apparaît `a` dans `t`.
   (b) Déterminer la complexité de la fonction `nb`.
2. (a) En déduire une fonction `elu1` prenant en argument un un tableau `t` d'entiers de $[\![0, k-1]\!]$, et renvoyant la valeur élue par `t` si elle existe, ou -1 si elle n'existe pas.
   (b) Déterminer la complexité de `elu1`, en fonction de k et N (la longueur de `t`).

Exercice 4 (Diviser pour régner simple) :

1. Soit T un tableau de longueur N, et soit $m = \lfloor N/2 \rfloor$.
   Démontrer que si a est élu par T, alors a est élu par $T[0:m]$ ou $T[m:N]$.
2. Écrire une fonction `nb2` prenant en arguments un tableau d'entiers `t`, un entier `a`, et deux indices `i` et `j`, et renvoyant le nombre de fois qu'apparait `a` dans `t[i:j]`.
3. (a) Écrire une fonction `dpr_simple` prenant en arguments un tableau `t` et deux indices `i` et `j`, et renvoyant :
   - $(-1, 0)$ si `t[i:j]` n'a pas de valeur élue ;
   - (a, q) si a est élu dans `t[i:j]`, et y apparait exactement q fois.

   Cette fonction devra appliquer le principe **diviser pour régner**.

   (b) Démontrer la terminaison de `dpr_simple`.
4. En déduire une fonction `elu2` prenant en argument un tableau `t` d'entiers et renvoyant la valeur élue par `t` si elle existe, ou -1 si elle n'existe pas.
5. Déterminer la complexité de la fonction `elu2`.

Exercice 5 (Diviser pour régner complexe) : Soit T un tableau de longueur N. On dit que $a \in \mathbb{N}$ est un **postulant** de T pour la valeur $n \in \mathbb{N}$ si :

- $n > N/2$;
- a apparait au plus n fois dans T ;
- tout entier $b \neq a$ apparait au plus $N - n$ fois dans T.

On dit que a est un postulant de T s'il existe n tel que a soit postulant de T pour n.

1. Montrer que 3 est un postulant de $T =$ `[|1;2;3;4;3;2;3;3|]`.
2. Montrer que si a est élu par T, alors a est un postulant de T.
3. Montrer que si a est un postulant de T, alors aucune autre valeur ne peut être élue par T.
4. (a) Donner un exemple de tableau ayant un postulant mais pas d'élu.

   (b) Donner une exemple de tableau n'ayant pas de postulant.
5. Soit T un tableau de longueur N **paire**. Soit $m = N/2$.

   (a) On suppose que $T[m:N]$ n'a pas d'élu, et que a est postulant de $T[0:m]$ pour une valeur l. Montrer que a est postulant de T pour une valeur n que l'on exprimera en fonction de m et l.

   (b) On suppose que a est un postulant de $T[0:m]$ pour l, et que b est un postulant de $T[m:N]$ pour q.

   i. On suppose que $a = b$. Montrer que a est un postulant de T pour une valeur de n que l'on exprimera en fonction de l et q.

   ii. On suppose que $a \neq b$ et $q > l$. Montrer que b est un postulant de T pour $n = m + q - l$.

   iii. On suppose que $a \neq b$ et $q = l$. Montrer que T n'admet pas d'élu.
6. Écrire une fonction `postulant` prenant en arguments un tableau d'entiers `t` et deux indices `i` et `j`, et renvoyant un couple (a, n) tel que :
   - si $(a, n) = (-1, 0)$, alors `t[i:j]` n'a pas d'élu ;
   - sinon, a est un postulant de `t[i:j]` pour n.

   On supposera pour simplifier que la longueur de `t[i:j]` est une puissance de 2.
   Cette fonction devra appliquer le principe diviser pour régner.
7. En déduire une fonction `elu3` prenant en argument un tableau `t` d'entiers et renvoyant la valeur élue par `t` si elle existe, ou -1 si elle n'existe pas.
8. Déterminer la complexité de `elu3`.