Exercice 1 : Miroir d'un tableau

Q. 1 Écrire une fonction `miroir_a_to_l : 'a array -> 'a list` prenant en paramètre un tableau et retournant le miroir du tableau, sous forme de liste. Par exemple `miroir [|0; 1; 2; 3; 4|] = [4; 3; 2; 1; 0]`.

Q. 2 Écrire une fonction `miroir_l_to_a : 'a list -> 'a array` prenant en paramètre une liste et retournant le miroir de cette liste, sous forme de tableau. Par exemple `miroir [0; 1; 2; 3; 4] = [|4; 3; 2; 1; 0|]`.

Exercice 2 : Transposition

On considère des tableaux de taille n de listes d'entiers de $[\![0, n-1]\!]$. On appelle *transposé* d'un tel tableau t, un tableau t' de taille n, contenant dans sa case $i \in [\![0, n-1]\!]$ une liste telle que $\forall j \in [\![0, n-1]\!], j \in t[i] \Leftrightarrow i \in t'[j]$.

Q. 1 Définir une fonction `transpose : int list array -> int list array` calculant la transposée du tableau qui lui est passé en argument.

Exercice 3 : Liste d'éléments à tableau indicateur

Soit une constante entière $M \in \mathbb{N}$ fixée ($M = 5$ dans les exemples). On considère deux représentations en OCaml des sous-ensembles de $[\![0, M-1]\!]$ (par exemple l'ensemble $\{0, 2, 3\}$).

a) La première au moyen d'une liste finie des éléments de l'ensemble (`[0; 2; 3]` ou `[3; 2; 0]` pour l'exemple ci-dessus).

b) La seconde au moyen d'un tableau de taille M de booléen dont la case $i \in [\![0, M-1]\!]$ indique si oui ou non l'élément est dans l'ensemble (`[|true; false; true; true; false|]` pour l'exemple ci-dessus).

Q. 1 Donner des fonctions de conversion `a_to_b : int -> int list -> bool array` et `b_to_a : bool array -> int list` d'un type vers l'autre. La fonction `a_to_b` prend en paramètres non seulement la liste en question mais aussi l'entier M de l'énoncé.

Exercice 4 : Polynômes creux

On considère dans cet exercice des polynômes de $\mathbb{Z}[X]$. On représente de tels polynômes en OCaml au moyen d'une liste de couples d'entiers : la liste `[(`a_0`, `d_0`); (`a_1`, `d_1`); (`a_2`, `d_2`); …; (`a_p`, `d_p`)]` avec pour tout $i \in [\![0, p]\!]$, $a_i \in \mathbb{Z}^\star$ et $d_i \in \mathbb{N}^\star$. Une telle liste représente alors le polynôme :

$$X^{d_0}(a_0 + X^{d_1}(a_1 + X^{d_2}(a_2 + \dots X^{d_{p-1}}(a_{p-1} + a_p X^{d_p}))))$$

On définit donc le type suivant.

```
type poly_1 = (int * int) list
```

On considère par ailleurs la représentation "classique" d'un polynôme $\sum_{i=0}^{n} b_i X^i$ au moyen d'un tableau de ses $n+1$ coefficients : `[|`b_0`; `b_1`; …; `b_n`|]`.
On définit donc le type suivant.

```
type poly_2 = int array
```

Q. 1 Donner des fonctions de conversion permettant la traduction d'une représentation vers l'autre. Discuter de la pertinence de l'une vis-à-vis de l'autre.

Exercice 5 : Cycle

Soit $n \in \mathbb{N}$. On dit d'une liste l d'entiers deux-à-deux distincts de $[\![0, n-1]\!]$ que c'est une liste *de cycle*. On dit qu'un tableau t' est le permuté d'un tableau t selon une liste de cycle $l =$ `[`l_0`; `l_1`; …; `l_{p-1}`]` lorsque t' est obtenu à partir de t en déplaçant en case $l_{(i+1) \mod p}$ l'élément se trouvant en case l_i, et ce, pour tout $i \in [\![1, p]\!]$.

Q. 1 Écrire une fonction `permute : int array -> int list -> unit` prenant en paramètres un tableau t de taille n et une liste de cycle l et modifiant le tableau en paramètre de sorte qu'il contienne le permuté du tableau t selon la liste l.

Q. 2 Définir une fonction `find_opt : (f: int -> bool) (n: int): int option` retournant le plus petit entier `i` de $[\![0, \texttt{n} - 1]\!]$ tel que `(f i)` vaut vrai. Si un tel indice n'existe pas, on retournera `None`.

Q. 3 Écrire une fonction `est_permute : int array -> int array -> int list` prenant en paramètres deux tableaux t et t' de taille n contenant des éléments deux à deux distincts et retournant une liste de cycle l telle que t' est le permuté de t selon la liste l. Si une telle liste n'existe pas, on lèvera un message d'erreur. *Indication : on s'autorisera à utiliser les fonctions du module `List`.*