TD #9 — Tableaux en OCaml, en C

Exercice 1. *Échauffement sur des* `'a array`*s*

1. Écrivez une fonction OCaml :
   `rev_array : 'a array -> 'a array`
   telle que `rev_array a` s'évalue en un `'a array` nouvellement construit dont les éléments sont ceux de `a` « en sens inverse ».
   Vous *devez* pour cela utiliser `Array.init`
2. Commentez l'intérêt d'une telle fonction, notamment en comparaison avec son homologue sur les `'a list`.
3. Écrivez une fonction OCaml :
   `is_palin : 'a array -> bool`
   qui s'évalue à `true` (resp. `false`) si son argument représente (resp. ne représente pas) un palindrome.
   Cette fonction *devra* utiliser `Array.for_all` (même si cela est un peu sous-optimal et en fait pas si pratique & joli).
4. Écrivez une nouvelle version de votre fonction qui utilise toujours `Array.for_all` mais ne fait pas de vérifications inutiles. (Le résultat n'est pas forcément beau ; en tout cas le mien est assez moche.)

Exercice 2. *Tableaux de tableaux* a la mano

On souhaite construire des tableaux à deux dimensions non rectangulaires : un tel tableau peut se représenter par un tableau de tableaux `a` dont les éléments sont eux-mêmes des tableaux. On adopte pour l'instant une représentation *en ligne* (« *row major* ») : l'indice `i` de `a` contient le tableau de tous les éléments de la $i$ème « ligne » du tableau.

1. Écrivez une fonction OCaml :
   `make_2D : int -> int list -> 'a -> 'a array array`
   telle que `make2D nrows rowdims e` s'évalue en un `'a array array` nouvellement créé dont tous les éléments sont initialisés à `e`, et possédant `nrows` lignes dont les dimensions (le nombre d'éléments) est donné par `rowdims` (dans « le même ordre »).
   Par exemple, on a :
   ```
   utop # make_2D 3 [0;1;2] ();;
   - : unit array array = [|[||]; [|()|]; [|(); ()|]|]
   ```
   Vous *pouvez* par exemple utiliser `List.iteri` si vous le souhaitez.
2. Écrivez une fonction C de signature :

`int **make_2D(size_t nrows, size_t rowdims[nrows], int e)`
qui fait de même *mutatis mutandis* pour un tableau de tableau d'`int` (de durée de stockage « allouée ») représenté par un pointeur de pointeur.
On pourra supposer que toutes les allocations mémoire se déroulent avec succès (que faudrait-il faire sinon ?).

3. Quelle différence y a-t'il entre les objets créés par les fonctions OCaml & C ci-dessus, en terme de métadonnées ? Qu'est-ce que cela implique pour leurs utilisations respectives ?
4. Proposez une fonction C `free_2D` permettant de libérer un objet créé par `make_2D`.

Exercice 3. *Linéarisation de tableaux multidimensionnels*

Dans cet exercice, on souhaite représenter un tableau multidimensionnel (disons à deux dimensions) par un tableau à une seule dimension. On suppose plus précisément que le tableau à représenter est rectangulaire, de n lignes et m colonnes. Dans une représentation linéarisée « en ligne » (*row major*) d'un tel tableau, les m premiers éléments sont ceux de la première ligne, les n suivants ceux de la seconde, etc.
Par exemple, le tableau à deux dimensions (ici un `int array array`) `[|[|0; 1|]; [|2; 3|]|]` est linéarisé (« en ligne ») en `[|0; 1; 2; 3|]`
On définit le type :

```
struct lin_2d_array
{
    size_t size;
    size_t colsize;
    int *dat;
};
```

dans le but de manipuler en C de tels tableaux linéarisés.

1. Écrivez une fonction C de signature :
   `int get(struct lin_2d_array arr, size_t i, size_t j)`
   qui renvoie la valeur de l'élément (i, j) du tableau à deux dimensions représenté (de façon linéarisée) par `arr`.
2. De même pour une fonction `set` de signature & spécifications adaptées.

Une représentation linéarisée « en colonne » (*column major*) est similaire à une représentation linéarisée « en ligne », où l'on a simplement que les n premiers éléments sont ceux de la première colonne etc. On peut remarquer que calculer une représentation en colonne d'un tableau à partir de sa représentation en ligne est équivalent à représenter sa transposée en ligne.

3. Écrivez une fonction C de signature :
   `struct lin_2d_array transpo(struct lin_2d_array arr)`
   réalisant cette opération.