Problème
Déplacement d'un cavalier sur un échiquier

> Pour toutes les fonctions récursives, **on essaiera, dans la mesure du possible, de proposer une implémentation récursive terminale**.
> Cependant, une version non terminale correcte sera toujours mieux notée qu'une version terminale incorrecte.
> En cas de doute sur la version terminale, je vous invite à écrire une version non terminale puis, à proposer en dessous votre tentative de version terminale, pour sécuriser l'obtention de points.

Un échiquier est un plateau avec 8 lignes et 8 colonnes. Ces lignes et ces colonnes seront dans cet exercice numérotées de 0 à 7 à partir du coin supérieur gauche. Une position sur l'échiquier sera un couple (i, j) d'entiers naturels avec i le numéro de ligne et j le numéro de colonne.
Un cavalier placé sur l'échiquier se déplace en bougeant de deux cases dans une direction et de une case perpendiculairement. On représente ci-dessous à titre d'exemple les positions que peut atteindre un cavalier en un déplacement (marquées par un X) à partir d'une position initiale (marquée par un C) :

	0	1	2	3	4	5	6	7
0								
1			X		X			
2		X				X		
3				C				
4		X				X		
5			X		X			
6								
7								

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4	X		X					
5				X				
6		C						
7				X				

Une position (i, j) sera dite :

- **Valide** si elle appartient bien à l'échiquier.
- n-**Accessible** à partir de la position (k, l), $n \in \mathbb{N}$, si elle est valide et si le cavalier peut atteindre la position (i, j) à partir de la position (k, l) en **au minimum** n déplacements. Dans le cas $n = 1$ on dira que (i, j) est un **successeur** de (k, l).
  Ainsi la seule position 0-accessible à partir de (i, j) est (i, j), et les croix des tableaux précédents représentent les successeurs de la position C.
- **Accessible** à partir de la position (k, l) si elle est valide et si le cavalier peut l'atteindre à partir de la position (k, l) en un nombre fini de déplacements.

1. **Recherche des successeurs.**
   On considère ici et dans tout le reste de l'exercice que l'on a accès à une liste `dep` contenant les déplacements autorisés du cavalier :

```
# let dep=[(-2,-1);(-2,1);(-1,-2);(-1,2);(1,-2);(1,2);(2,-1);(2,1)];;
val dep : (int * int) list =
  [(-2, -1); (-2, 1); (-1, -2); (-1, 2); (1, -2); (1, 2); (2, -1); (2, 1)]
```

Où par exemple $(-1, 2)$ signifie que le cavalier passe (si c'est possible) d'une position (i, j) à une position $(i-1, j+2)$.

(a) Écrire une fonction `valide` de type `int * int -> bool` qui vérifie si une position est valide.

(b) Écrire une fonction `successeurs (i,j)` qui retourne la liste des successeurs de (i, j). Elle devra retourner la liste vide si (i, j) n'est pas valide.

(c) Écrire une fonction `flat` qui transforme en liste une liste de listes. Un résultat possible est par exemple :

```
# flat [ [] ; [(1,1)] ; [(1,1);(2,2)] ; [(1,1);(2,2);(3,3)] ];;
- : (int * int) list = [(1, 1); (1, 1); (2, 2); (1, 1); (2, 2); (3, 3)]
```

L'ordre des couples dans le résultat n'a aucune importance et ne doit pas nécessairement suivre celui de cet exemple.

(d) Écrire une fonction `liste_successeurs l` qui retourne la liste de tous les successeurs de toutes les positions d'une liste de positions `l`. Elle devra retourner la liste vide si `l` est vide et on ne cherchera pas à éliminer les doublons éventuels.

2. **Construction de la matrice d'accès.**
Dans toute cette partie (i, j) désignera une position valide quelconque.
Pour tout $n \in \mathbb{N}$, on note $M(n)$ la matrice de dimension 8x8 dont on numérote les lignes et les colonnes de 0 à 7 pour respecter la convention de l'échiquier et dont le coefficient en k^{eme} ligne et l^{eme} colonne vaut :

- p si la position (k, l) est p-accessible à partir de la position (i, j) avec $p \leq n$.
- -1 sinon.

On a par exemple à partir de la position $(0, 0)$ (en remplaçant les -1 par des $*$) :

$$M(0) = \begin{pmatrix} 0 & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{pmatrix}, \ M(1) = \begin{pmatrix} 0 & * & * & * & * & * & * & * \\ * & * & 1 & * & * & * & * & * \\ * & 1 & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{pmatrix}$$

$$M(2) = \begin{pmatrix} 0 & * & 2 & * & 2 & * & * & * \\ * & * & 1 & 2 & * & * & * & * \\ 2 & 1 & * & * & 2 & * & * & * \\ * & 2 & * & 2 & * & * & * & * \\ 2 & * & 2 & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{pmatrix}$$

Enfin on note M la matrice de dimension 8x8 dont le coefficient en k^{eme} ligne et l^{eme} colonne (numérotées de 0 à 7) vaut p si la position (k, l) est p-accessible à partir de la position (i, j), -1 sinon.

Voici quelques commandes permettant de créer/modifier des matrices (qui sont en fait des tableaux de tableaux) en OCaml :

- On crée une matrice de dimension nxp dont les coefficients sont tous égaux à *expr* avec l'instruction `Array.make_matrix n p` *expr* :

```
# let m=Array.make_matrix 2 4 (-1);;
val m : int array array = [|[|-1; -1; -1; -1|]; [|-1; -1; -1; -1|]|]
```

- On accède à l'élément de la matrice `m` situé en ligne `n` et en colonne `p` avec `m.(n).(p)`.
- On modifie l'élément de la matrice `m` situé en ligne `n` et en colonne `p` avec `m.(n).(p)<-`*expr*.

```
# m.(1).(3)<-8;;
- : unit = ()
# m;;
- : int array array = [|[|-1; -1; -1; -1|]; [|-1; -1; -1; 8|]|]
```

(a) Prouver qu'il existe un entier $N \in \mathbb{N}$ tel que pour toute position valide (i, j) et pour tout $n \in \mathbb{N}$, $n \geq N \Rightarrow M(n) = M$.

(b) Écrire une fonction `transition l m p` qui, lorsque `l` est la liste des successeurs des positions $(p-1)$-accessibles à partir d'une position initiale et lorsque `m` est la matrice $M(p-1)$, retourne la liste des positions p-accessibles et modifie `m` afin qu'elle soit égale à $M(p)$.

(c) Écrire une fonction `cavalier (i,j)` qui retourne la matrice M obtenue à partir de la position initiale (i, j). On supposera que (i, j) est une position valide.

(d) Justifier que `cavalier` se termine toujours en temps fini.

3. **Recherche de positions inaccessibles.**
On cherche à savoir si toute position est accessible à partir de toute position initiale. Plutôt que chercher à le prouver mathématiquement (pas très difficile mais pénible à cause d'un certain nombre de cas particuliers), on va le vérifier à l'aide d'un programme.
Pour cette question et uniquement pour cette question, on pourra utiliser des instructions `for` ou `while`.

Écrire une fonction `test` de type `unit -> bool` qui retourne `true` s'il existe une position inaccessible à partir d'au moins une position initiale, `false` sinon. *Après avoir proposé une approche exhaustive, on pourra proposer, en justifiant par des arguments mathématiques, une approche permettant de limiter le nombre de tests effectués.*