★ **Exercice 1.** À cheval ! (8 points)

Dans cet exercice, vous devrez écrire un certain nombre de fonctions python qui suivent chacune une signature imposée (nom de fonction et types des paramètres). **Toutes vos fonctions devront être implémentées dans un même fichier python dénommé :** `P1.py`. Vous êtes autorisés (et il est même conseillé) d'écrire d'autres fonctions annexes.

Sur un échiquier, le cavalier se déplace de façon singulière en L. Pour cela il se déplace de façon horizontale ou verticale de deux cases puis d'une case (ou d'une case puis de deux) de façon perpendiculaire.
On suppose que l'échiquier est modélisé par une liste dénommée `echiquier` de taille $n \times n$ d'entiers. `echiquier[i]==0` si le cavalier n'est pas passé (au sens ne n'est jamais posé après un déplacement) dans la ligne l et la colonne c qui correspondent à cet indice, sinon la cellule vaut 1.

1	0	0	0
0	0	1	0
0	1	0	0
0	0	0	1

Figure 1: Exemple de progression d'un cavalier sur l'échiquier et de son incidence sur les valeurs des cases

▷ **Question 1.** (0,5 pt) Écrivez une fonction qui renvoie l'indice associé à la case dans la liste `echiquier` :

```
indice_de_position(position: Tuple[int, int], taille_cote: int) -> int
```

Le numéro de la première ligne et de la première colonne est 1 et le contenu associé est à l'indice 0 de la liste `echiquier`. Le paramètre `taille_cote` représente la longueur d'un côté de l'échiquier.

▷ **Question 2.** (1 pt) Écrivez une fonction :

```
suivants(position: Tuple[int, int], taille_cote: int) -> List[Tuple[int, int]]
```

qui, étant donnée une position courante du cavalier (en paramètre de la fonction) et la longueur du côté de l'échiquier (`taille_cote`), calcule et retourne toutes les positions atteignables sur l'échiquier par celui-ci en un déplacement.

▷ **Question 3.** (4 pts) Étant donné un échiquier de longueur de côté `taille_cote` donné et une position initiale du cavalier sous forme d'indice de ligne et d'indice de colonne, concevez une fonction récursive qui recherche un parcours pour le cavalier à l'issue duquel il est passé dans toutes les cases de l'échiquier sans passer deux fois dans la même case.

Le profil attendu de la fonction qui permet de réaliser cela est le suivant :

```
cavalier(echiquier: List[int], position: Tuple[int, int], taille_cote: int) -> bool
```

Ses paramètres sont :

- l'échiquier (liste de taille n^2 d'entiers dont toutes les valeurs sont initialement à 0),
- la position du cavalier (tuple indiquant la ligne et la colonne) où il s'est posé (l'échiquier passé en paramètre a donc une valeur non nulle à la position correspondante),
- la longueur d'un côté de l'échiquier.

Le résultat renvoyé par la fonction est :

- une valeur booléenne qui est égale à `True` si une solution au problème a été trouvée, à `False` sinon.

▷ **Question 4.** (3 pts) Étendez votre fonction précédente pour qu'elle renvoie, lorsqu'une solution a été trouvée, le chemin parcouru par le cavalier depuis sa position de départ. Votre nouvelle fonction aura le profil suivant :

```
chemin_cavalier(echiquier: List[int], position: Tuple[int, int], taille_cote: int)
    -> Tuple[bool, List[Tuple[int, int]]]
```

Ses paramètres sont:

- l'échiquier (liste de taille n^2 d'entiers),
- la position du cavalier (tuple indiquant la ligne et la colonne) où il s'est posé (l'échiquier passé en paramètre a donc une valeur non nulle à la position correspondante),
- la longueur d'un côté de l'échiquier.

Le résultat renvoyé par la fonction est un tuple composé de :