

Centres étrangers – 2025 – sujet1 - Correction

Exercice 2 (6 points)

Question 1

Réponse : **6**

👉 *Commentaire : L'indice [1][2] correspond à la 2e ligne (indice 1) et à la 3e colonne (indice 2), donc 6.*

Question 2

```
def plateau_init(n, m, cartes):  
    shuffle(cartes)  
    plateau = []  
    for i in range(n):  
        plateau.append([cartes[i+j*n] for j in range(m)])  
    return plateau
```

👉 *Commentaire : on suit le squelette : la ligne 5 est donnée. Les indices $i + j \times n$ ($i < n, j < m$) sont tous différents et compris entre 0 et $n \times m - 1$: chaque carte n'est utilisée qu'une fois (tirage sans remise).*

Question 3

```
def cartes_voisines(n, m, i, j):  
    voisines = []  
    for i2 in range(i-1, i+2):  
        for j2 in range(j-1, j+2):  
            if (i2, j2) != (i, j) and \  
                i2 in range(n) and \  
                j2 in range(m):  
                voisines.append((i2, j2))  
    return voisines
```

👉 *Commentaire : On parcourt les 8 directions autour d'une case, en vérifiant que les coordonnées sont dans la grille.*

Question 4

e1 : $9 + 8 + 6 + 7 = \mathbf{30}$ (cartes bien voisines deux à deux).

e2 : **ce n'est pas une chaîne** : (1,2) et (2,0) ne sont pas voisines.

e3 : **n'existe pas** : (1,4) est hors du plateau (les colonnes vont de 0 à 3).

👉 *Commentaire : Il faut additionner les valeurs aux coordonnées indiquées dans le plateau, si elles existent.*

Question 5

```
def chaine_evalue(plateau, chaine):  
    s = 0
```

```

    for i, j in chaine:
        s = s + plateau[i][j]
    return s

```

👉 *Commentaire : On parcourt les coordonnées de la chaîne pour sommer les valeurs dans le plateau.*

Question 6

Le parcours en largeur explore d'abord les chemins les plus courts.

👉 *Commentaire : Cela permet de trouver la première solution valide avec le minimum de cartes, ce qui correspond à l'objectif du jeu.*

Question 7

```

def explore(plateau, cible):
    n, m = len(plateau), len(plateau[0])
    a_visiter = file_init()
    for i in range(n):
        for j in range(m):
            file_ajoute(a_visiter, [(i,j)])

    while not file_est_vide(a_visiter):
        chemin = file_retre(a_visiter)
        if chaine_evalue(plateau, chemin) == cible:
            return chemin
        dernier = chemin[-1]
        i0, j0 = dernier
        for i, j in cartes_voisines(n, m, i0, j0):
            if (i, j) not in chemin:
                file_ajoute(a_visiter,
                           chemin + [ (i,j) ])

    # Pas de solutions
    return None

```

👉 *Commentaire : On utilise une file pour faire un parcours en largeur à partir de chaque case, en ajoutant des voisins valides.*

👉 *Commentaire : avec le plateau de l'énoncé, explore(plateau, 27) renvoie [(0, 0), (0, 1), (1, 1)], c'est-à-dire 6 + 7 + 14.*

Question 8

Au lieu de retourner le premier chemin trouvé, il faut stocker tous les chemins valides dans une liste et continuer l'exploration.

On crée une liste vide `solutions` au début ; on remplace `return chemin` par `solutions.append(chemin)` et on continue le parcours ; à la fin, on renvoie `solutions` au lieu de `None`.