

DS 2 Terminale NSI - Correction

Exercice 1

Partie A

Q1

- attribut : itinéraire
- méthode : remplir_grille()

Q2

- a = 4
- b = 7

Q3

```
def remplir_grille(self):
    i, j = 0, 0 # Position initiale
    self.grille[j][i] = 'S' # Case de départ marquée d'un S
    for direction in self.itineraire:
        if direction == 'D':
            i = i + 1 # Déplacement vers la droite
        elif direction == 'B':
            j = j + 1 # Déplacement vers le bas
        self.grille[j][i] = '*' # Marquer le chemin avec '*'
    self.grille[j][i] = 'E' # Case d'arrivée marquée d'un E
```

Q4

```
def get_dimensions(self) -> tuple:
    """
    @return longueur et largeur de l'itinéraire
    """
    return self.longueur, self.largeur
```

Q5

```
def remplir_grille(self):
    i, j = 0, 0 # Position initiale
    self.grille[j][i] = 'S' # Case de départ marquée d'un S
    for direction in self.itineraire:
        if direction == 'D':
            i = i + 1 # Déplacement vers la droite
        elif direction == 'B':
            j = j + 1 # Déplacement vers le bas
        self.grille[j][i] = '*' # Marquer le chemin avec '*'
    self.grille[j][i] = 'E' # Case d'arrivée marquée d'un E
```

Partie B

Q6

```
def itineraire_aleatoire(m : int, n : int) -> str:
    """
    @param m -- longueur de l'itinéraire
    @param n -- largeur de l'itinéraire
    @return l'itinéraire
    """
    itineraire = ""
    i, j = 0, 0
    while i != m and j != n:
        deplacement = choice(['D', 'B'])
        itineraire += deplacement
        if deplacement == 'D':
            j += 1
        else:
            i += 1
    if i == m:
        itineraire = itineraire + 'D'*(n-j)
    if j == n:
        itineraire = itineraire + 'B'*(m-i)
    return itineraire
```

Partie C

Q7

Pour une grille de 1 dimension, il n'y a qu'un seul chemin possible : aller toujours à droite ('D') ou toujours vers le bas ('B').

- exemple, pour 1x3 : "DDD".
- exemple, pour 3x1 : "BBB".

Q8

Chaque chemin vers la case (m, n) provient de (m-1, n) : un mouvement vers le bas 'B', ou de (m, n-1) : un mouvement vers la droite 'D'.

Donc, le nombre de chemins vers (m, n) est la somme du nombre de chemins vers (m-1, n) + du nombre de chemins vers (m, n-1).

Q9

```
def nombre_chemins(m : int, n : int) -> str:
    """
    @param m -- longueur de l'itinéraire
    @param n -- largeur de l'itinéraire
    @return nombre de chemins possibles
    """
    if m == 1 or n == 1:
        return 1
    return nombre_chemins(m-1, n) + nombre_chemins(m, n-1)
```

Exercice 2

Partie A

Q1

11 = 1011₂ : “chef d’équipe” + “conducteur” + “équipier”

Q2

chef d’agrès + chef d’équipe + conducteur + équipier : $2^2 + 2^1 + 2^3 + 2^0 = 15 = 1111_2$

Q3

cela correspond à l’aptitude “chef d’agrès”, or un chef d’agrès est nécessairement un chef d’équipe et un équipier et il faut lui ajouter ces aptitudes dans son codage.

Q4

$2^8 - 15 - 1 = 240$

Q5

- “équipier” : 8 octets, soit $1 - 1/8 = 87,5 \%$
- “équipier” + “chef d’équipe” + “chef d’agrès” + “conducteur” : 44 octets, $1 - 1/44 = 97,7 \%$

Partie B

Q6

- une clé primaire identifie de manière unique chaque enregistrement dans une table
- une clé étrangère établit une relation entre deux tables en faisant référence à la clé primaire d'une autre table.

Q7

La valeur de la clé étrangère idagres n’existe pas dans la table agres

Q8

UPDATE intervention

SET heure = ‘10:44:06’

WHERE id = 3

Q9

Charlot

Red

Kevin

Q10

SELECT nom

FROM personnel

WHERE qualif >= 16 AND actif = 1;

Q11

- Nombre de chefs d'agrès prêt à intervenir le 27 mars 2024. (2)
- Nombre d'interventions de chefs d'agrès prêt à intervenir le 27 mars 2024. (1)

Q12

```
SELECT DISTINCT personnel.nom  
FROM personnel  
INNER JOIN agres ON personnel.matricule =  
agres.idchefagres  
WHERE agres.jour = '2024-02-15';
```

Q13

```
SELECT DISTINCT personnel.nom  
FROM personnel  
INNER JOIN agres ON personnel.matricule =  
agres.idchefagres  
INNER JOIN moyen ON agres.id =  
moyen.idagres  
INNER JOIN intervention ON moyen.idinter =  
intervention.id  
WHERE intervention.jour = '2024-06-11';
```