2024 centres étrangers groupe 1 jour 2

Exercice 1

Partie A

1. `self.longueur` est un attribut et `remplir_grille` est une méthode.

2. `a = 4` et `b = 7`

3.

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

4.

```
def get_dimensions(self):
    return self.longueur, self.largeur
```

5.

```
def tracer_chemin(self):
    for i in range(self.largeur + 1):
        for j in range(self.longueur + 1):
            # on remplace les "." par des " "
            if self.grille[i][j] == ".":
                print(" ", end="")
            else:
                print(self.grille[i][j], end="")
        print()
```

Partie B

6.

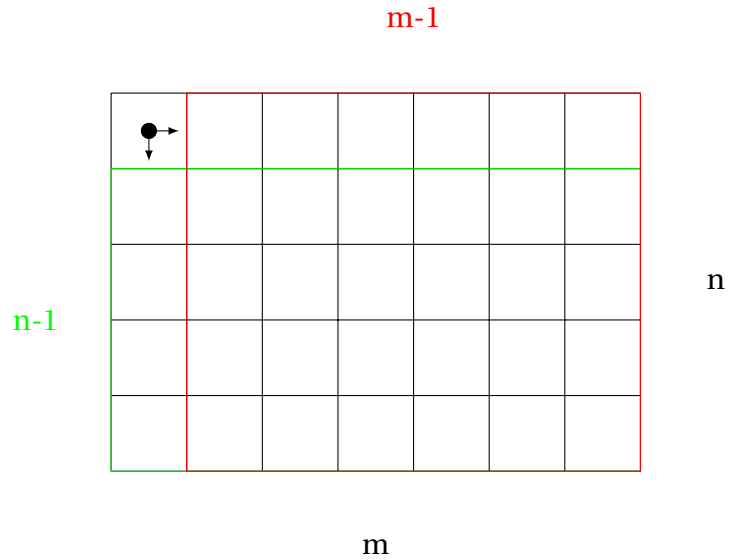
```
from random import choice

def itineraire_aleatoire(m, n):
    itineraire = ''
    i, j = 0, 0
    while i != m and j != n
        deplacement = random.choice(['D', 'B'])
        itineraire = itineraire + deplacement
        if deplacement == "D":
            j = j + 1
        else:
            i = i + 1
        if i == m:
            itineraire = itineraire + 'D'*(n-j)
        if j == n:
            itineraire = itineraire + 'B'*(m-i)
    return itineraire
```

Partie C

7. Le seul chemin possible est en ligne droite vers le bas n fois. Par exemple si $n = 4$ l'itinéraire est BBBB.

8.



Pour le premier déplacement il n'y a que deux possibilités, à droite ou en bas. Si on va à droite on se retrouve dans une grille (rouge) $(m-1) \times n$ et donc il y a $N(m-1,n)$ chemins. Si on va en bas on se retrouve dans une grille (verte) $m \times (n-1)$ et donc il y a $N(m,n-1)$ chemins. Comme les chemins de ces deux cas sont distincts (les cases de départ sont différentes) on a alors :

$$N(m,n) = N(m-1,n) + N(m,n-1)$$

9.

```
def nombre_chemins(m, n):
    if m == 1 or n == 1:
```

```
        return 1
    else:
        return nombre_chemins(m-1, n) + nombre_chemins(m, n-1)
```

Exercice 2

Partie A

1. `ls documents`

2. Le dossier `multimedia` et son contenu va être déplacé dans le dossier `documents`.

3. Le code correspond à un arbre binaire car chaque nœud a au maximum deux fils. Or l'arborescence possède un nœud (documents) avec trois fils.

4. C'est un parcours préfixe.

5. On suppose que le déplacement de la question 2 n'a pas eu lieu :
home - documents - multimedia - cours - administratif - personnel - images - videos - films

Partie B

6.

```
def est_vide(self):
    return self.fils == []
```

7.

```
var_films = Dossier("films", [])
var_videos = Dossier("videos", [var_films])
var_images = Dossier("images", [])
var_multimedia = Dossier("multimedia", [var_images, var_videos])
```

8.

```
def parcours(self):
    print(self.nom)
    for f in self.fils:
        f.parcours()
```

9. Le nombre de fichiers est fini et il n'y a pas de cycle dans un arbre, donc cette méthode se terminera toujours.

10.

```
def parcours(self):
    for f in self.fils:
        f.parcours()
    print(self.nom)
```

11. `ls` n'affiche que les enfants d'un nœud alors que la méthode `parcours` affiche tous les descendants.

12.

```
def mkdir(self, nom):
    dos = Dossier(nom, [])
    self.fils.append(dos)
```

13.

```
def contient(self, nom_dossier):
    file = []
    for dossier in self.fils:
        if dossier.nom == nom_dossier:
            return True
        file.append(dossier)
    while file != []:
        dossier_parent = file.pop(0)
        for dossier in dossier_parent:
            if dossier.nom == nom_dossier:
                return True
            file.append(dossier)
    return False
```

14.
On peut utiliser la même structure que la méthode `contient` sauf qu'il faudrait renvoyer le nom du parent au lieu de renvoyer `True` quand on trouve le bon nom.
[Voilà une implémentation, même si ça n'est pas demandé :]

```
def parent(self, nom_dossier):
    file = []
    for dossier in self.fils:
        if dossier.nom == nom_dossier:
            return self.nom
        file.append(dossier)
    while file != []:
        dossier_parent = file.pop(0)
        for dossier in dossier_parent:
            if dossier.nom == nom_dossier:
                return dossier_parent.nom
            file.append(dossier)
    return False
```

15. On pourrait ajouter un attribut `self.parent` à la classe `Dossier`.

Exercice 3

Partie A

1. Le chef d'équipe correspond à 2^1, il est aussi équipier avec le code 2^0 et conducteur avec le code 2^3. On a donc comme qualification :

$$2^3 + 2^1 + 2^0 = 8 + 2 + 1 = 3$$

2. Le chef d'agrès conducteur possède toutes les aptitudes :

$$2^3 + 2^2 + 2^1 + 2^0 = 8 + 4 + 2 + 1 = 15$$

3. La qualification 4 correspond à uniquement chef d'agrès. Or un chef d'agrès est aussi nécessairement chef d'équipe et équipier, il faut donc lui ajouter 2 et 1. La valeur 4 est donc impossible.

4. Sur 8 bits, il reste encore 4 bits. Il est donc possible de coder encore 4 nouvelles aptitudes.

5. On peut avoir à coder les quatre aptitudes soit à peu près 40 caractères. On utiliserait donc 40 octets. On économise donc 39 octets sur 40 soit $39 * 100/40 = 98\%$.

Partie B

6. Une clé primaire est un champ qui permet d'identifier de manière unique une ligne dans une table. Une clé étrangère fait référence à la clé primaire d'une autre table. Elle n'est pas forcément unique.

7. `idagres` est une clé étrangère qui fait le lien avec la clé primaire `id` de la table `agres`. Or la table `agres` n'a pas de ligne avec un `id` de 5.

8.

```sql
INSERT INTO intervention SET heure = '10:44:06' WHERE id = 3;
```

9.

nom
Charlot
Red
Kevin

10.
[Il ne semble pas y avoir de correspondance avec la qualif de la partie précédente !]

```sql
SELECT nom
FROM personnel
WHERE actif = 1 AND qualif >= 16;
```

11.
Requête A :

count(*)
2

Elle affiche le nombre d'agrès du 27 mars 2024.
Requête B :

count(*)
1

Elle affiche le nombre d'agrès du 27 mars 2024 ayant eu une intervention.

12.

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

13.

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