Centres étrangers – 2024 – sujet2 - Correction

Exercice 1 (6 points)

---

1. Donner un attribut et une méthode de la classe `Chemin`

Un attribut possible est `itineraire` (la chaîne de caractères représentant le chemin).
Une méthode possible est `remplir_grille()` ou `tracer_chemin()`.

---

2. Valeurs des variables `a` et `b`

Itinéraire : `"DDBDBBDDDDB"`

Nombre de D = 7
Nombre de B = 4

👉*Commentaire :*
`a = 4` *(largeur = nombre de B)*
`b = 7` *(longueur = nombre de D)*

---

3. Méthode `remplir_grille`

```
def remplir_grille(self):
    i = 0
    j = 0
    self.grille[i][j] = "*"
    for lettre in self.itineraire:
        if lettre == "D":
            j += 1
        else:  # B
            i += 1
        self.grille[i][j] = "*"
```

👉*Commentaire : On part de la case (0,0). À chaque déplacement, on modifie les coordonnées puis on marque la case visitée avec* `*`.

---

4. Méthode `get_dimensions`

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

👉*Commentaire : La méthode renvoie un tuple contenant la longueur (nombre de D) et la largueur (nombre de B).*

---

5. Méthode `tracer_chemin`

```
def tracer_chemin(self):
    for ligne in self.grille:
        print(" ".join(ligne))
```

👉*Commentaire : On affiche chaque ligne de la grille proprement pour visualiser le chemin.*

---

6. Génération aléatoire d'un chemin

```
def chemin_aleatoire(m, n):
    itineraire = ""
    i = 0
    j = 0
    while i < n and j < m:
        deplacement = random.choice(["D", "B"])
        itineraire += deplacement

        if deplacement == "D":
            j += 1
        else:
            i += 1
    itineraire += "D" * (m - j)
    itineraire += "B" * (n - i)
    return itineraire
```

👉*Commentaire : On tire au sort tant qu'on n'a pas atteint la dernière ligne ou colonne, puis on complète obligatoirement pour atteindre l'arrivée.*

---

7. Justifier que N(1, n) = 1 et N(m, 1) = 1

Si une dimension vaut 1, on ne peut se déplacer que dans **une seule direction.** Il n'existe donc qu'**un seul chemin** possible.

---

8. Justifier la relation de recurrence

Lorsqu'on se trouve sur la case de coordonnées (m,n), pour arriver en bas à droite :

- soit on vient de la case au-dessus: case **(m,n−1)**
- soit on vient de la case à gauche: **(m-1,n)**

On additionne donc les deux possibilités: **N(m,n)=N(m−1,n)+N(m,n−1)**

---

9. Fonction récursive `nombre_chemins`

```python
def nombre_chemins(m, n):
    if m == 1 or n == 1:
        return 1
    return nombre_chemins(m-1, n) + nombre_chemins(m, n-1)
```

👉*Commentaire : Si une dimension vaut 1, il n'y a qu'un seul chemin. Sinon, on applique la relation de récurrence montrée dans la question précédente.*
