SUJET 0 – 2024 – sujet1 - Correction

Exercice 2 (6 points)

---

1. Fonction `corrige`

```
def corrige(cop, corr):
    resultat = []
    for i in range(len(cop)):
        resultat.append(cop[i] == corr[i])
    return resultat
```

👉 **Commentaire :** *On compare chaque réponse donnée avec la correction et on ajoute True ou False dans la liste résultat.*

---

2. Fonction `note` (sans liste auxiliaire)

```
def note(cop, corr):
    compteur = 0
    for i in range(len(cop)):
        if cop[i] == corr[i]:
            compteur += 1
    return compteur
```

👉 **Commentaire :** *On compte directement les bonnes réponses sans construire de liste intermédiaire.*

---

3. Fonction `notes_paquet`

```
def notes_paquet(p, corr):
    resultat = {}
    for nom in p:
        resultat[nom] = note(p[nom], corr)
    return resultat
```

👉 **Commentaire :** *On parcourt chaque candidat du dictionnaire et on associe sa note dans un nouveau dictionnaire.*

---

4. Peut-on utiliser une liste comme clé ?

Non.

👉 **Commentaire :** *Une liste n'est pas « hashable » (elle est modifiable), donc elle ne peut pas être utilisée comme clé de dictionnaire. En revanche, un tuple (couple) est immuable et peut servir de clé.*

---

5. Autre solution pour éviter les homonymes

On peut utiliser un **identifiant unique** (numéro candidat, identifiant interne).

👉 **Commentaire :** *Un identifiant numérique anonyme protège mieux les données personnelles et évite les conflits d'homonymes.*

Partie : Fonction `enigme`

Dictionnaire donné :

```
{('Tom','Matt'):6,
 ('Lambert','Ginne'):4,
 ('Carl','Roth'):2,
 ('Kurt','Jett'):4,
 ('Ayet','Finzerb'):3}
```

---

6. Résultat de `enigme`

La fonction conserve les **trois meilleures notes** dans a, b, c.

Résultat :

```
(
 (('Tom','Matt'), 6),
 (('Lambert','Ginne'), 4),
 (('Kurt','Jett'), 4),
 {('Carl','Roth'):2, ('Ayet','Finzerb'):3}
)
```

👉 **Commentaire :** *a = meilleur, b = deuxième, c = troisième ; d contient tous les autres.*

---

7. Que calcule la fonction ?

Elle sépare :

- les **trois meilleures notes**
- et les autres candidats

👉 **Commentaire :** *Elle effectue un tri partiel en conservant le top 3.*

---

8. Cas avec moins de 3 entrées

Si moins de 3 candidats :

- certaines variables parmi a, b, c restent à None.

👉 **Commentaire :** *La fonction renvoie quand même un tuple (a, b, c, d), mais certains éléments peuvent valoir None.*

---

9. Fonction `classement`

```
def classement(notes):
    resultat = []
    a, b, c, d = enigme(notes)
    if a is not None:
        resultat.append(a)
    if b is not None:
        resultat.append(b)
    if c is not None:
        resultat.append(c)
    while d:
        a, b, c, d = enigme(d)
        if a is not None:
            resultat.append(a)
        if b is not None:
            resultat.append(b)
        if c is not None:
            resultat.append(c)
    return resultat
```

👉 **Commentaire :** *On applique successivement enigme pour récupérer les notes par ordre décroissant.*

Partie : renote_express

10. Compléter `renote_express2`

```
def renote_express2(copcorr):
    gauche = 0
    droite = len(copcorr)
    while droite - gauche > 1:
        milieu = (gauche + droite)//2
        if copcorr[milieu]:
            gauche = milieu
        else:
            droite = milieu
    if copcorr[gauche]:
        return gauche + 1
    else:
        return gauche
```

👉 **Commentaire :** *On applique une recherche dichotomique du premier False.*

---

11. Coût en temps

- `renote_express` :
  👉 coût **O(n)**
- `renote_express2` :
  👉 coût **O(log n)**

👉 **Commentaire :** *La recherche dichotomique divise l'intervalle par 2 à chaque étape.*

---

12. Adaptation pour corriger directement une copie

Au lieu de créer la liste de booléens :

On teste directement les réponses avec la correction et on applique une recherche dichotomique sur les indices.

Principe :

- vérifier si la réponse au milieu est correcte
- adapter gauche/droite selon le résultat

👉 **Commentaire :** *On évite ainsi de construire la liste complète et on obtient une correction en temps logarithmique.*
