Centres étrangers – 2023 – sujet2 - Correction

Exercice 3 (4 points)

---

1. Fonction `bonjour`

```
def bonjour(nom):
    return 'Bonjour ' + nom + ' !'
```

a. Résultat de `bonjour('Alan')`

```
'Bonjour Alan !'
```

👉 *Commentaire : L'opérateur + concatène les chaînes de caractères.*

---

b. Valeurs de x et y

Programme :

```
une_chaine = 'Bonjour'
x = (une_chaine[2] == une_chaine[3])
y = (une_chaine[4] == une_chaine[1])
```

Indices :

Indice	Lettre
0	B
1	o
2	n
3	j
4	o

Donc :

- `x = ('n' == 'j')` → False
- `y = ('o' == 'o')` → True

Résultat :

- x : booléen False

- y : booléen True

  👉 *Commentaire : Une comparaison renvoie toujours un booléen.*

---

c. Fonction `occurrences_lettre`

```
def occurrences_lettre(une_chaine, une_lettre):
    compteur = 0
    for caractere in une_chaine:
        if caractere == une_lettre:
            compteur += 1
    return compteur
```

👉 *Commentaire : On parcourt la chaîne caractère par caractère. Les chaînes sont immuables mais parcourables.*

---

2. Arbres Binaires de Recherche (ABR)

Liste :

```
animaux = ['python', 'chameau', 'pingouin', 'renard', 'gnou']
```

Ordre alphabétique :

```
chameau < gnou < pingouin < python < renard
```

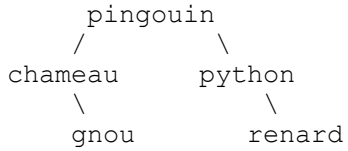
---

a. ABR de hauteur minimale

Pour minimiser la hauteur, on choisit l'élément médian :

Racine : pingouin

Arbre :



👉 *Commentaire : On équilibre autant que possible pour réduire la hauteur.*

b. ABR de hauteur maximale

On insère dans l'ordre alphabétique croissant :

```
chameau
   \
    gnou
       \
        pingouin
             \
              python
                   \
                    renard
```

👉 *Commentaire : On obtient un arbre dégénéré (structure en liste chaînée).*

---

3. Fonction `mystere`

```
def mystere(un_abr):

    if un_abr.est_vide():
        return 0
    else:
        return 1 + mystere(un_abr.sous_arbre_gauche) \
                 + mystere(un_abr.sous_arbre_droit)
```

a. Résultat

La fonction compte :

- 1 pour chaque nœud
  - taille du sous-arbre gauche
  - taille du sous-arbre droit

Donc : `mystere(abr_mots_francais)`

renvoie **336531**

👉 *Commentaire : La fonction calcule la taille de l'arbre. Cet algorithme récursif classique est à connaître.*

---

b. Fonction `hauteur`

```
def hauteur(un_abr):
    if un_abr.est_vide():
        return 0
    else:
        return 1 + max(
            hauteur(un_abr.sous_arbre_gauche),
            hauteur(un_abr.sous_arbre_droit)
        )
```

👉 *Commentaire : La hauteur correspond au nombre de niveaux. On prend la taille maximum des deux sous-arbres.*

---

4. Recherche de mots croisés

a. Compléter la ligne 4

```
def chercher_mots(liste_mots, longueur, lettre, position):
    res = []
    for i in range(len(liste_mots)):
        if len(liste_mots[i]) == longueur and liste_mots[i][position] ==
lettre:
            res.append(liste_mots[i])
    return res
```

👉 *Commentaire : On vérifie d'abord la longueur pour éviter une erreur d'indice.*

---

b. Double appel

Commande :

```
chercher_mots(
    chercher_mots(liste_mots_francais,3,'x',2),
3,'a',1
)
```

Interprétation :

1. On sélectionne les mots de 3 lettres avec 'x' en position 2.
2. Parmi ces mots, on garde ceux de 3 lettres avec 'a' en position 1.

👉 *Commentaire :*

- *Cela filtre successivement les mots répondant aux deux contraintes.*
- *On obtient les mots de forme _ a x.*

---

c. Mot de 5 lettres finissant par "ter"

"ter" correspond aux positions 2, 3 et 4.

Commande :

```
chercher_mots(
    chercher_mots(
        chercher_mots(liste_mots_francais,5,'t',2),
        5,'e',3),
    5,'r',4)
```

👉 *Commentaire : On applique successivement les contraintes sur chaque position.*
