Exercice 4 (4 points)**.**

Cet exercice porte sur la notion de chaînes de caractères, de tableau et sur la programmation de base en Python.

On appelle palindrome un texte dont l'ordre des lettres reste le même, qu'on le lise de la droite vers la gauche ou de la gauche vers la droite.
Par exemple, un mot à une lettre est un palindrome, `"BOB"` est un palindrome, tout comme `"LAVAL"`. Le mot `""` est considéré comme un palindrome.
On souhaite programmer une fonction `palindrome` qui prend en paramètre une chaîne de caractères `txt`. Cette fonction renvoie `True` si la chaîne de caractères `txt` est un palindrome, `False` sinon.

On rappelle que :

— La fonction `len` prend une chaîne de caractères en paramètre et renvoie sa longueur, c'est-à-dire le nombre de lettres constituant la chaîne de caractères.
Exemple : `len("bonjour")` vaut 7.

— Si `txt` est une chaîne de caractères, `txt[i]` est la lettre de `txt` d'indice i. Attention, la première lettre a pour indice 0.
Exemple : si `txt = "bonjour"` alors `txt[2]` désigne `"n"`.

1. On donne ci-dessous une implémentation récursive incomplète pour la fonction `palindrome`.
L'opération `+` à la ligne 8 permet de concaténer deux chaînes de caractères.
Exemple : Si `txt1 = "bon"` et `txt2 = "jour"`, l'instruction `txt1 + txt2` renvoie la chaîne de caractères `"bonjour"`.

```
1 def palindrome(txt):
2     """ Str -> Bool """
3     ----------------:
4     ----------------
5     taille = len(txt)
6     interieur = ""
7     for i in range(1, taille - 1):
8         interieur = interieur + txt[i]
9     return (txt[0] == txt[taille - 1]) and (palindrome(interieur))
```

(a) Choisir parmi les propositions ci-dessous celle qui qui convient pour compléter la fonction palindrome (ligne 3 et 4) :

Proposition 1 :

```
1 if len(txt) <= 2:
2     return True
```

Proposition 3 :

```
1 if len(txt) < 2:
2     return True
```

Proposition 2 :

```
1 if len(txt) < 2:
2     return False
```

Proposition 4 :

```
1 if len(txt) <= 2:
2     return False
```

(b) Lors de l'appel de `palindrome("bonjour")`, indiquer les valeurs, à la ligne 9, de : `txt[0]`, `txt[taille - 1]` et `interieur`.

2. Proposer deux tests pour cette fonction qui permettent de tester deux cas de figure différents en justifiant ce choix.
3. Écrire une version non récursive de la fonction `palindrome`.
4. On étudie dans cet exercice des chaînes de caractères utilisant uniquement les lettres `"A"`, `"T"`, `"C"` et `"G"`.

   Exemple : `"AA"`, `"CAT"` et `"CCGATACG"`.

   On associe à chacune de ces lettres une autre lettre appelée lettre complémentaire selon le tableau suivant :

Lettre	`"A"`	`"T"`	`"G"`	`"C"`
Lettre complémentaire	`"T"`	`"A"`	`"C"`	`"G"`

   On obtient le complémentaire d'un mot en remplaçant chacune de ses lettres par sa lettre complémentaire.

   Exemple : Le complémentaire à `"GAATTC"` est `"CTTAAG"`.

   (a) Écrire une fonction en Python, nommée `complementaire`, qui prend en paramètre une chaîne de caractères `txt` écrite uniquement avec les lettres `"A"`, `"C"`, `"G"` et `"T"`. Cette fonction renvoie la chaîne de caractères complémentaire de `txt`.

   Exemple : l'appel `complementaire("GAATTC")` renvoie `"CTTAAG"`.

   (b) Une chaîne de caractères `txt` est dite palindromique si la concaténation de `txt` avec son complémentaire est un palindrome.

   Exemples :

   `"GAATTC"` est palindromique car `"GAATTC" + "CTTAAG" = "GAATTCCTTAAG"` est un palindrome.

   `"GAAT"` n'est pas palindromique car `"GAAT" + "CTTA" = "GAATCTTA"` n'est pas un palindrome.

   Déterminer si la chaîne de caractères `"GATCGT"` est palindromique.

   (c) Écrire une fonction `est_palindromique` prenant comme paramètre une chaîne de caractères `txt`. Cette fonction doit renvoyer `True` s'il s'agit d'une séquence palindromique, `False` sinon.