



22-NSIJ1JA1 : Corrigé

Année : **2022**

Centre : **Asie-Pacifique**

Jour : **1**

Enoncé :



1. Exercice 1

algorithmique, chaînes de caractères, complexité

1.	Initialisation		Etape 1	Etape 2	Etape 3	Etape 4
	$\neg (i \leq j)$		Vrai	Vrai	Vrai	Faux
	$(\text{mot}[i] \neq \text{mot}[j])$		Faux	Faux	Faux	
	i	0	1	2	3	3
	j	4	3	2	1	1
	p	vrai	vrai	vrai	vrai	vrai

2. 3 comparaisons

2.b

- si n est pair on a $n/2$ comparaisons,
- si n est impair on a $n/2 + 1$ comparaisons.

1. Avant la première itération $(j - i)$ est égale à $\text{longueur}(\text{mot}) - 1$. $(i - j)$ est donc strictement positif pour tout mot constitué de plus d'une lettre .

À chaque tour de boucle j diminue d'une unité ($j = j - 1$) et i augmente d'une unité ($i = i + 1$), nous avons donc $(j - i)$ qui diminue de deux unités à chaque tour de boucle, $(j - i)$ est donc décroissante. Après un certains nombres de tours de boucle $(j - i)$ va donc être égale à zéro (nous aurons alors $i = j$).

Pour le tour de boucle suivant, (après que j ait été décrémenté d'une unité et i ait été incrémenté d'une unité), nous aurons $i > j$ ce qui provoquera l'arrêt de la boucle.

Nous pouvons donc affirmer que la boucle se termine.

2. La boucle est exécutée 4 fois.

Il est possible de modifier l'algorithme comme suit :

Texte

```
Fonction palindrome2(mot):  
  Variables : i,j : ENTIER ; p : BOOLEEN  
  i ← 0  
  j ← longueur(mot)-1  
  tant que i ≤ j  
    Si mot[i] ≠ mot[j]  
      Renvoie Faux  
    FinSi  
    i ← i+1  
    j ← j-1  
  Fin tant que  
  Renvoie Vrai
```

Avec cette modification, la 1^{re} fois où $\text{mot}[i] \neq \text{mot}[j]$ la fonction renvoie FAUX et l'algorithme s'arrête (il est inutile d'examiner les autres lettres). Avec l'exemple du mot "routeur", nous aurions effectué 1 tour de boucle complet et nous serions sortis de la fonction dès le début du 2^e tour de boucle.

2. Exercice 2

base de données

1.a.

|attribut|type| |:---:|:---:| | id_plat|INT| | nom_plat|VARCHAR(100)| | prix_plat| VARCHAR(100)|

1.b

| relation | clé primaire | |:---:|:---:| | plat | id_plat | | tablea_salle | num_table | | client | num_client | |
reservation | num_reservation |

1.c On trouve 2 clés étrangères dans la relation reservation : num_table et num_client. Une clé étrangère permet de créer une jointure entre 2 relations.

2.a.

Requête SQL

```
SELECT nom_plat, type_plat, prix_plat  
FROM plat
```

2.b.

Requête SQL

```
SELECT nom_plat
FROM plat
WHERE type_plat = 'Dessert'
```

2.c.

Requête SQL

```
UPDATE plat
SET tel_client = "0602030405"
WHERE num_client = 42
```

2.d.

Requête SQL

```
SELECT nom_client
FROM client JOIN reservation ON reservation.num_client = client.num_client
WHERE reservation.num_table = 13
```

3. Exercice 3

systèmes d'exploitation

1.a `cd ../projet`

1.b `cd /home/sam/projet`

2.a `ls ./projet`

2.b `chmod u+w ./projet/config.txt`

3.a l'option `r` permet de supprimer le répertoire ciblé par cette commande, mais aussi les répertoires et fichiers contenus dans ce répertoire cible.

3.b le système d'exploitation a réalisé, pour effacer ces dossiers et fichiers, un parcours en profondeur de l'arbre.

1. L'appel de cette fonction renvoie 1 (renvoie le nombre de fichiers dont le nom commence par la lettre `b` minuscule) :
2. 1er appel ($i = 0$) on considère le fichier `'nsi.bmp'`, le nom ne commence pas par `'b'`, on appelle `nb_fichiers(list_fich, i+1)`.
3. 2e appel ($i=1$) on considère le fichier `'banana.mp3'`, le nom commence par `'b'`, on appelle $1 + \text{nb_fichiers}(\text{list_fich}, i+1)$.
4. 3e appel ($i = 2$) on considère le fichier `'job.txt'`, le nom ne commence pas par `'b'`, on appelle `nb_fichiers(list_fich, i+1)`.
5. 4e appel ($i = 3$) on considère le fichier `'BoyerMoore.py'`, le nom ne commence pas par `'b'`, on appelle `nb_fichiers(list_fich, i+1)`.

6. 5e appel (i = 4), on a i == len(list_fich) qui est vraie, on arrête les appels récursifs.

4. Exercice 4

programmation objet en langage Python

1.

 **Script Python**

```
def ajouter_beurre(self, qt):  
    self.qt_beurre = self.qt_beurre + qt
```

2.

 **Script Python**

```
def afficher(self):  
    print("farine : "+str(self.qt_farine))  
    print("beurre : "+str(self.qt_beurre))  
    print("oeufs : "+str(self.nb_oeufs))
```

3.

 **Script Python**

```
def stock_suffisant_brioche(self):  
    return self.qt_beurre >= 175 and self.qt_farine >= 350 and self.nb_oeufs >= 4
```

4.a. La valeur affichée dans la console est 2. Il est possible de fabriquer 2 brioches avec le stock actuel.

4.b

farine : 300

beurre : 650

oeufs : 2

1.

 **Script Python**

```
def nb_brioches(liste_stocks):  
    nb = 0  
    for s in liste_stocks:  
        nb = nb + s.produire()  
    return nb
```

5. Exercice 5

programmation Python

1.a. La fonction renvoie [2, 6].

1.b. La fonction mystere renvoie les coordonnées du personnage après avoir parcouru le chemin passé en paramètre de la fonction (et en partant de l'origine du repère).

1.

Script Python

```
def accessible(dep, arrivee):  
    arr = mystere(dep)  
    return arr[0]==arrivee[0] and arr[1]==arrivee[1]
```

2.

Script Python

```
from random import randint  
  
def chemin(arrivee):  
    deplacement = '00000000'  
    while not accessible(deplacement, arrivee) :  
        deplacement=""  
        for k in range(8):  
            pas = str(randint(0,1))  
            deplacement = deplacement + pas  
    return deplacement
```

3. La plus grande valeur en binaire qui permet d'atteindre le point [5, 3] est 11100000 (il faut que les bits de poids fort soit à 1, on commence donc par monter avant de commencer à se déplacer vers la droite). En base 10 cela donne 224.