Centres étrangers - juin 2021 - sujet 1 (corrigé)

Exercice 1 (Algorithmique et chaînes de caractères)

1. On a le tableau suivant :

	initialisation	étape 1	étape 2	étape 3	étape 4
`i ⩽ j`		`Vrai`	`Vrai`	`Vrai`	`Faux`
`mot[i] ≠ mot[j]`		`Faux`	`Faux`	`Faux`	
`i`	`0`	`1`	`2`	`3`	`3`
`j`	`4`	`3`	`2`	`1`	`1`
`p`	`Vrai`	`Vrai`	`Vrai`	`Vrai`	`Vrai`

2. (a) Il y a trois comparaisons.

   (b) Si n est pair on a $n/2$ comparaisons et si n est impair on a $n/2+1$ comparaisons, où $/$ représente la division entière.

3. Avant la première itération `(j - i)` est égale à `longueur(mot)-1`. Ainsi, `(i-j)` est donc strictement positif pour tout mot constitué de plus d'une lettre. A chaque tour de boucle, `j` diminue d'une unité (on a `j = j - 1`) et `i` augmente d'une unité (on a `i = i +1`), et donc `(j-i)` diminue de deux unités. La quantité `(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` a été décrémenté d'une unité et `i` incrémenté d'une unité), on aura `i > j` ce qui provoquera l'arrêt de la boucle. On peut donc affirmer que la boucle se termine.

4. La boucle est exécutée quatre fois. Il est possible de modifier l'algorithme comme suit :

```
fonction palindrome2(mot) :
  variables : i,j : ENTIER ; p : BOOLEEN
  i ← 0
  j ← longueur(mot) - 1
  tant que i ⩽ j
    si mot[i] ≠ mot[j] alors
      renvoyer Faux
    fin si
    i ← i + 1
    j ← j - 1
  fin tant que
  renvoyer Vrai
fin fonction
```

Avec cette modification, la première fois où `mot[i] ≠ 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"`, on aurait effectué un seul tour de boucle complet et on serait sorti de la fonction dès le début du deuxième tour de boucle.

Exercice 2 (Bases de données et SQL)

1. (a) On a les types suivants :

attribut	type
id_plat	INT
nom_plat	VARCHAR(100)
type_plat	VARCHAR(100)
prix_plat	FLOAT

(b) On a les clés primaires suivantes :

relation	clé primaire
plat	id_plat
table_salle	num_table
client	num_client
reservation	num_reserv

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

2. (a) La requête suivante convient :

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

(b) La requête suivante convient :

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

(c) La requête suivante convient :

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

(d) La requête suivante convient :

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

Exercice 3 (Systèmes d'exploitation)

1. (a) Il faut saisir la commande : `cd ../projet`
   (b) Il faut saisir la commande : `cd /home/sam/projet`
2. (a) Il faut saisir la commande : `ls ./projet`
   (b) Il faut saisir la commande : `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.
   (b) Le système d'exploitation a réalisé, pour effacer ces dossiers et fichiers, un parcours en profondeur de l'arbre.
4. L'appel de cette fonction renvoie le nombre de fichiers dont le nom commence par la lettre `b` minuscule, c'est-à-dire `1` :
   - ⋆ Premier appel (`i = 0`) : on considère le fichier `'nsi.bmp'`. Comme le nom ne commence pas par `'b'`, on appelle `nb_fichiers(list_fich, i+1)`.
   - ⋆ Deuxième appel (`i=1`) : on considère le fichier `'banana.mp3'`. Comme le nom commence par `'b'`, on appelle `1 + nb_fichiers(list_fich, i+1)`.
   - ⋆ Troisième appel (`i = 2`) : on considère le fichier `'job.txt'`. Comme le nom ne commence pas par `'b'`, on appelle `nb_fichiers(list_fich, i+1)`.
   - ⋆ Quatrième appel (`i = 3`) : on considère le fichier `'BoyerMoore.py'`. Comme le nom ne commence pas par `'b'`, on appelle `nb_fichiers(list_fich, i+1)`.
   - ⋆ Cinquième appel (`i = 4`) : on a `i == len(list_fich)` qui est vraie, donc on arrête les appels récursifs.

Exercice 4 (POO)

1. La fonction suivante convient :

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

2. La fonction suivante convient :

```
>>> mon_stock = Stock()
>>> mon_stock.afficher()
farine: 0
oeuf: 0
beurre: 0
>>> mon_stock.ajouter_beurre(560)
>>> mon_stock.afficher()
farine: 0
oeuf: 0
beurre: 560
```

3. La fonction suivante convient :

```
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 donc possible de fabriquer deux brioches avec le stock actuel.

   (b) L'affichage console est le suivant :

```
def produire(self):
    res = 0
    while self.stock_suffisant_brioche():
        self.qt_beurre = self.qt_beurre - 175
        self.qt_farine = self.qt_farine - 350
        self.nb_oeufs = self.nb_oeufs - 4
        res = res + 1
    return res
```

5. La fonction suivante convient :

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

Exercice 5 (Programmation générale)

1. (a) La fonction renvoie `[2, 6]`.

   (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).

2. La fonction suivante convient :

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

3. La fonction suivante convient :

```
from random import randint
def chemin(arrivee):
    deplacement = '00000000'
    while ..... :
        .....
        for k in range(8):
            pas = str(randint(0,1))
            ..... = deplacement + .....
    return deplacement
```

4. 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.