Amérique du sud - septembre 2022 - sujet 1 (corrigé)

Exercice 1 (Bases de données)

1. ⋆ L'attribut `idMere` ne peut pas être pris comme clé primaire puisqu'une mère peut avoir plusieurs enfants.
   ⋆ Le couple `(date, rang)` peut être pris comme clé primaire puisque la date identifie l'année et le mois de naissance, puis le rang identifie le bébé dans le mois. Il est donc unique.
   ⋆ Le couple `(poids, taille)` ne peut pas être pris comme clé primaire puisque plusieurs bébés peuvent naître avec le même poids et la même taille.
2. Cette requête lève une erreur puisque l'attribut `idMere` est clé étrangère de la table `Naissances` faisant référence à l'attribut `numPatiente` et qu'il existe encore des entrées `Naissances` attachées à cette mère (le bébé Samson Pauline).
3. La requête suivante convient :

```
INSERT INTO Patientes VALUES (13862, 'Bélanger', 'Ninette', 'La Rochelle')
```

4. La requête suivante convient :

```
UPDATE Naissances SET prenom = 'Laurette' WHERE idMere = 13860 AND prenom = 'Lorette'
```

5. La requête suivante convient :

```
SELECT nom, prenom FROM Patientes WHERE commune = "Aigrefeuille d'Aunis"
```

6. La requête suivante convient :

```
SELECT AVG(poids)
FROM Naissances
JOIN TypesAccouchement ON idAcc = acc
WHERE libelleAcc = 'césarienne'
```

7. Cette requête renvoie le nom et le prénom des mères dont les bébés sont nés avec le type d'accouchement numéro 1 (la voie naturelle si l'ordre d'affichage est respecté), à savoir :

Berthelot	Michelle
Samson	Marine
Baugé	Gaëlle
Baugé	Gaëlle

Exercice 2 (Programmation et tri)

1. On écrit : `attente.append((50,4))`
2. (a) L'algorithme proposé est un tri en place par sélection.
   `ATTENTION : malgré ce qui est indiqué dans l'énoncé, cette fonction ne renvoie pas la liste triée (elle renvoie None), elle modifie la liste passée en paramètre à la fonctio`
   (b) La complexité en temps des tris par insertion et sélection est quadratique selon la taille du tableau, c'est-à-dire en $O(n^2)$.
3. (a) La fonction suivante convient :

```
def quitte(attente)
   return [attente[i] for i in range(1,len(attente))] # attente[1:] fonctionnerait aussi
```

   (b) La fonction suivante convient :

```
def maj(attente):
   return [(c[0], c[1]-1) for c in attente]
```

4. (a) La fonction suivante convient :

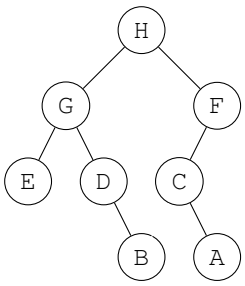
```
def priorite(attente, p):
   for i in range(len(attente)):
      if attente[i][0] == p:
         return attente[i][1]
   return -1
# si le patient n'existe pas
```

   (b) La fonction suivante convient :

```
def revise(attente, p):
   nouvelle = []
   n = priorite(attente, p)
   for (patient, prio) in attente :
      if patient == p:
         nouvelle.append((patient, 1))
      elif prio < n:
         nouvelle.append((patient, prio + 1))
      else : # patients ayant déjà un nombre prioritaire plus élevé
         nouvelle.append((patient, prio))
   return nouvelle
```

Exercice 3 (Arbres binaires)

1. L'arbre a pour hauteur 5 et est de taille 11.
2. (a) La structure implémentée correspond à l'arbre 2.
   (b) On a l'arbre suivant :



3. (a) Il s'agit d'un parcours en postfixé. Ainsi l'affichage est le suivant

```
d
b
g
f
a
```

   (b) La fonction suivante convient :

```
def parcours_maladies(arb):
    if arb == {}:
        return None
    parcours_maladies(arb['sag'])
    parcours_maladies(arb['sad'])
    if len(arb['sag'])==0 and len(arb['sad'])==0:
        print(arb['etiquette'])
```

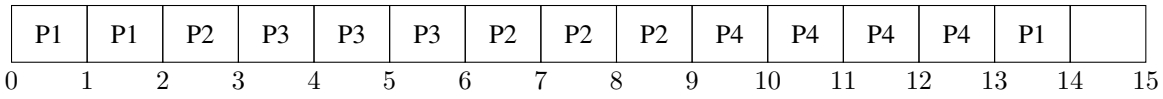
4. La fonction suivante convient :

```
def symptomes(arb, mal):
    if arb['sag'] != {}:
        symptomes(arb['sag'], mal)
    if arb['sad'] != {}:
        symptomes(arb['sad'], mal)
    if arb['etiquette'] == mal:
        arb['surChemin'] = True
        print('symptômes de', arb['etiquette'], ':')
    else:
        if arb['sad'] != {} and arb['sad']['surChemin']:
            print(arb['etiquette'])
            arb['surChemin'] = True
        if arb['sag'] != {} and arb['sag']['surChemin']:
            print( 'pas de ', arb['etiquette'] )
            arb['surChemin'] = True
```

Exercice 4 (Processus et OS)

A. Ordonnancement des processus.

1. On a le diagramme suivant :



2. On a le tableau suivant :

Processus	**Temps d'exécution**	**Instant d'arrivée**	**Temps de séjour**	**Temps d'attente**
P1	3	0	$14-0=14$	$14-3=11$
P2	4	2	$9-2=7$	$7-4=3$
P3	3	3	$6-3=3$	$3-3=0$
P4	4	5	$13-5=8$	$8-4=4$

3. Le temps d'attente d'un processus peut être nul si celui-ci est le processus le plus prioritaire durant toutes la durée de son exécution.

B. Processus et ressources.

1. On a la configuration suivante :
   - ⋆ L'analyseur d'échantillon attend D4 car D4 est utilisée par le SGBD
   - ⋆ Le SGBD attend D5 car D5 est utilisée par le tableur
   - ⋆ Le traitement de texte attend D3 car D3 est utilisée par le tableur
   - ⋆ Le tableur attend D1 car D1 est utilisée par le tableur

   Les processus s'attendent donc tous mutuellement.
2. Il s'agit d'une situation d'interblocage (deadlock en anglais).
3. L'ordre suivant est possible :
   - ⋆ Tableur (parce que D1 est libre)
   - ⋆ Traitement de texte (parce que le tableur a libéré D3)
   - ⋆ SGBD (car le tableur a libéré D5)
   - ⋆ Analyseur d'échantillon (car le SGBD a libéré D4)

Exercice 5 (Réseaux et protocoles de routage)

A. Adressage.

1. Le service de radiologie (RL R) possède pour adresse `192.168.1.0` et pour masque `/24` (soit `255.255.255.0`).
2. Les interfaces du routeur R5 ont pour adresses :
   - ⋆ avec R1 : `175.89.50.254`
   - ⋆ avec R4 : `44.197.5.1`
   - ⋆ avec le serveur patient : `192.168.5.254`
3. (a) Sur un réseau, les adresses extrêmes sont utilisées : 0 pour l'adresse du réseau et l'extrême supérieur pour la diffusion à l'ensemble du réseau (par exemple si le masque est /24, une l'adresse du broadcast réseau se termine par 255). Ains sur RL R, la première adresse IP utilisable est `192.168.1.1` et la dernière `192.168.1.254`.

   (b) Ainsi, on peut utiliser $254 - 1 + 1 = 254$ adresses sur le réseau RL R (dont 4 sont déjà utilisées).

B. Etude du protocole RIP (Routing Information Protocol).

1. Pour minimiser le nombre de sauts, le message suivra les routeurs : R5 – R1 – R0.
2. Si R1 est déconnecté, le paquet suivra les routeurs : R5 – R4 – R2 – R0 (ou en remplaçant R2 par R3).

C. Protocole OSPF (Open Shortest Path First).

1. La liaison R2-R3 possède un poids de
$$\frac{10^9}{400 \times 10^6} = \frac{10}{4} = 2{,}5$$
donc $C = 3$.
2. Une bande passante entre R3-R4 est $\dfrac{10^9}{5} = 2 \times 10^8$, soit 200 Mb/s.
3. On minimise la somme des poids du chemin parcouru : R5 – R4 – R2 – R1 – R0 de poids total 8.
4. De même, si le routeur R1 n'est plus accessible, le chemin deviendra R5 – R4 – R3 – R0 de poids total 10.