Correction 26-NSIJ2G11

Exercice 1

Partie A

1. ```
   from donnees import *
   from chauffeur import *
   from client import *
   from course import *
   from taxi import *
   ```

2. ```
   johndoe = Chauffeur("Doe", "John", "140159320012")
   ```

3. ```
   taxi1 = Taxi("HG-818-AV", "standard", "électrique", False)
   taxi1.choix_chauffeur(johndoe)
   ```

4. ```
   assert taxi.est_libre(), "Le taxi doit être libre pour pouvoir créer une course"
   ```

5. ```
   taxi.modifier_libre(False)
   ```

6. ```
   jeannedoe = Client("Doe", "Jeanne", "6 rue des ordinateurs, Paris, France", 2)
   course1 = Course(jeannedoe, taxi1, "6 rue des ordinateurs, Paris, France", "211 avenue
   Jean Jaurès, Paris, France")
   ```

7. ```
     def temps(self) -> float:
         return (self.date_arrivee - self.date_depart).total_seconds() / 60
   ```

8. ```
     def tarif(self):
         type_vehicule = self.taxi.type_vehicule
         donnees_tarifaire = vehicules[type_vehicule]
         prise_en_charge = donnees_tarifaire['prise_en_charge']
         tarif_h = donnees_tarifaire['tarif_horaire']
         prix_km = donnees_tarifaire['prix_km']
         tarif = prise_en_charge
         tarif = tarif + prix_km * self.distance()
         tarif = tarif + tarif_h * self.temps() / 60
         return tarif
   ```

Partie B

9. Dans le chiffrement symétrique, la même clé est utilisée pour chiffrer et déchiffrer les données, tandis que dans le chiffrement asymétrique, deux clés différentes sont utilisées : une clé publique pour chiffrer les données et une clé privée pour les déchiffrer.

10. Le problème de sécurité est que si la clé de session est interceptée par un attaquant lors de sa transmission, l'attaquant pourra déchiffrer tous les messages échangés entre l'application et le chauffeur, comme le chiffrement utilisé est symétrique.

11. Une stratégie possible pour pallier à ce problème est d'utiliser un chiffrement asymétrique pour échanger la clé de session de manière sécurisée. Par exemple, l'application pourrait chiffrer la clé de session avec la clé publique du chauffeur, de sorte que seul le chauffeur, qui possède la clé privée correspondante, puisse déchiffrer la clé de session et l'utiliser pour chiffrer les messages échangés par la suite.

    Une autre possibilité est d'utiliser un protocole de négociation de clé sécurisé, comme celui de Diffie-Hellman, pour permettre à l'application et au chauffeur de générer une clé de session partagée sans jamais la transmettre directement, réduisant ainsi le risque d'interception.

Antoine Dequay

Exercice 2

Partie A

1. ```
   SELECT modele FROM ordinateur
   WHERE etat = "Réparation";
   ```

2. ```
   INSERT INTO ordinateur
   VALUES (22, "Thomson", "MO5", "Panne");
   ```

3. ```
   UPDATE ordinateur
   SET etat = "Fonctionnel"
   WHERE id_ordi = 9;
   ```

4. Le résultat sera :

   2

Partie B

5. Les schémas relations sont :
   - `jeu(#id_jeu, titre, genre, etat, id_plat)`
   - `plateforme(#id_plat, nom, bits)`

6. ```
   SELECT titre FROM jeu
   JOIN plateforme ON jeu.id_plat = plateforme.id_plat
   WHERE nom = 'Oric';
   ```

7. ```
   SELECT titre FROM jeu
   JOIN plateforme ON jeu.id_plat = plateforme.id_plat
   JOIN ordinateur ON plateforme.id_plat = ordinateur.id_plat
   WHERE modele = 'Amstrad 6128';
   ```

Partie C

8. `ping 192.168.208.11`.

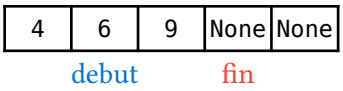
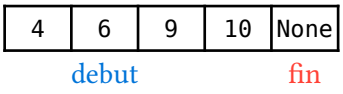
9. En binaire, 240 s'écrit 11110000. Ainsi, 4 bits sont alloués pour les hôtes (correspondant aux 4 derniers bits des adresses IP, à 0 sur le masque), ce qui permet d'avoir 2^4 - 2 - 3 = **11 adresses hôtes encore disponibles** (on soustrait 2 pour les adresses réservées et 3 pour les hôtes déjà présents : les deux ordinateurs et le routeur R8).

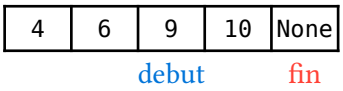
10. On a la table suivante :

Routeur R8		
Destination	Routeur suivant	nombre de sauts
R1	R1	3
R2	R5	2
R3	R2	3
R4	R4	1
R5	R5	1
R6	R7	2
R7	R7	1

11. Un chemin possible est : Ordinateur de Alan –> Switch R1 –> R8 –> R7 –> R6 –> Ordinateur de Ada.

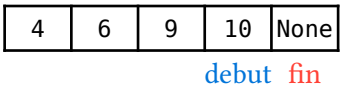
12. Le chemin parcouru est : R8 –> R5 –> R6 –> R1 –> R3, avec un coût total de 10 + 1 + 10 + 10 = **31**.

Antoine Dequay
   ```

Exercice 3

Partie A

1. C'est Von Neumann.
2. On trouve :
   1) Mémoire RAM
   2) Processeur
   3) Unité de contrôle
   4) Unité arithmétique et logique
3. Dire que la RAM est volatile signifie que les données stockées dans la RAM sont perdues lorsque l'ordinateur est éteint ou redémarré (lorsque la RAM n'est plus alimentée).
4. On trouve :
   1) Registre
   2) Mémoire cache
   3) Mémoire vive
   4) Disque dur
5. `sub r2, r5, r4`
6. Comme 0 n'est pas un nom de registre, l'instruction `add r1 r2 0` est invalide. Si on avait écrit `addi r1 r2 0`, alors on aurait copié la valeur de r2 dans r1.
7. ```
   addi r3 r1 0
   addi r1 r2 0
   addi r2 r3 0
   ```
8. On a :
   - **A** = 0
   - **B** = 1
   - **C** = 1
   - **D** = 1
9. On remarque que le circuit logique correspond à l'addition des trois entrées sous forme binaire, sous la forme **Rs S**.

Partie B

10. Le terme correspondant est FIFO, pour "First In, First Out".
11. `[None, None, None, None, None]`
12. On a :
    a)

    

    –> 4 renvoyé

    b)

    

    c)
   ```



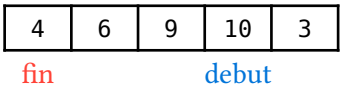
−> 6 renvoyé

d)

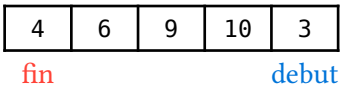


−> 9 renvoyé

e)

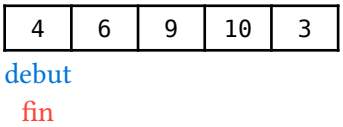


f)



−> 10 renvoyé

g)



−> 3 renvoyé

13. On a à la fin des instructions : `['o', 's', 'i', 'n', 'f']`.

14.
```
def est_vide(self):
    return self.debut == self.fin
```

15.
```
def retirer_element(self):
    res = self.contenu[self.debut]
    self.debut = self.debut + 1
    if self.debut == self.capacite:
        self.debut = 0
    return res
```