# Centres étrangers - juin 2026 - sujet 2 (corrigé)

**Exercice 1 (Programmation orienté objet, sécurité des communications)**
**Partie A**

1. Il faut ajouter les lignes suivantes dans le script `application.py` :

```
from donnees import vehicules, energie
from chauffeur import Chauffeur
from taxi import Taxi
from client import Client
from course import Course
```

2. Il faut ajouter les lignes suivantes dans le script `application.py` :

```
chauffeur0 = Chauffeur('Doe', 'John', '140159320012')
```

3. Il faut ajouter les lignes suivantes dans le script `application.py` :

```
taxi0 = Taxi('HG-818-AV', 'standard', 'électrique', False)
taxi0.choix_chauffeur(chauffeur0)
```

4. Il faut ajouter la ligne suivante au début du constructeur `__init__` de la classe `Course` :

```
assert taxi.est_libre()
```

5. Il faut ajouter la ligne suivante au constructeur de la classe `Course` :

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

6. Il faut ajouter les lignes suivantes dans le script `application.py` :

```
client0 = Client('Doe', 'Jeanne', '6 rue des ordinateurs, Paris', 2)
course0 = Course(client0, taxi0, '6 rue des ordinateurs, Paris',
                 '211 avenue Jean Jaurès, Paris')
```

7. Il faut écrire les lignes suivantes dans la classe `Course` :

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

8. Il faut écrire les lignes suivantes dans la classe `Course` :

```
def tarif(self): #Q8
        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. Un protocole de chiffrement utilise un procédé de chiffrement associé à une clé de chiffrement et un procédé de déchiffrement associé à une clé de déchiffrement. Dans le cas d'un algorithme symétrique, les clés de chiffrement et de déchiffrement sont identiques. Dans le cas d'un algorithme asymétrique, les clés de chiffrement sont différentes.
10. Un agent qui intercepterait la transmission contenant la clé de sessions auraient accès à cette clé. Il pourrait alors :

- Déchiffrer les communications entre le client et le chauffeur (après interception des messages) pouvant obtenir des informations privées.
- Usurper l'identité du chauffeur en communiquant directement avec le client.
- Usurper l'identité du client en communiquant directement avec le chauffeur.

11. On utilisera la stratégie suivante qui repose sur le chiffrement asymétrique. Le chauffeur possède une clé publique PUB et une clé privée PRI (par exemple avec le protocole RSA. La clé publique PUB est envoyé en clair à la centrale. La centrale encode la clé de session SES en utilisant la clé publique PUB et transmet le message chiffré obtenu [SES]PUB au chauffeur. Le chauffeur déchiffre le message en utilisant sa clé privé PRI. Il obtient ainsi le message SES.
Le chiffrement asymétrique est utilisé uniquement pour la transmission de la clé de session. On garde le chiffrement symétrique (avec cette clé de session) pour le reste des données on ne ralentit pas les échanges entre la centrale et le chauffeur.
La transmission de la clé est bien sécurisée car un agent extérieur ne peut pas déchiffrer le message [SES]PUB contenant la clé de session (chiffrée) puisque la clé privée n'a jamais transité dans la communication.

**Exercice 2 (Bases de données relationnelles, requêtes SQL, réseaux, protocoles de routage)**

1. On propose la requête suivante :

```
SELECT modele FROM ordinateur WHERE etat =  Réparation ;
```

2. On propose la requête suivante :

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

3. On propose la requête suivante :

```
UPDATE ordinateur SET etat =  Fonctionnel  WHERE id_ordi = 9;
```

4. Cela donne la table suivante qui indique le nombre d'ordinateurs de la marque Commodore en état fonctionnel.

| COUNT(*) |
|---|
| 2 |

**Partie B**

5. Le schéma relationnel avec les domaines des relations `jeu` et `plateforme` sont :
`jeu (id_jeu:INT, titre:TEXT, genre:TEXT, etat:TEXT, id_plat:INT)`
`plateforme (id_plat:INT, nom:TEXT, bit:INT)`
6. On propose la requête suivante :

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

7. On propose la requête suivante :

```
SELECT titre FROM jeu
JOIN ordinateur ON jeu.id_plat = ordinateur.id_plat
WHERE ordinateur.modele =  Armstrad 6128 ;
```

**Partie C**

8. Alan doit utiliser la commande `ping`
9. 240 en binaire s'écrit `11110000`.
Dans le masque de sous-réseau, seuls les 4 derniers bits sont à 0, c'est-à-dire qu'il y a $2^4 = 16$ adresses disponibles dans le sous-réseau. Il y a une adresse réservée pour le broadcast, une pour le routeur, et les deux adresses de Alan et Grace. Il reste $16 - 4 = 12$ adresses disponibles. Donc il est encore possible d'ajouter 12 machines à ce réseau local.
10. La table de routage complète est la suivante :

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

11. Le routeur associé à l'ordinateur d'Alan est R8. Celui associé à l'ordinateur de Ada est R6. D'après la table de routage, le routeur suivant pour aller de R8 à R6 est R7. Le chemin complet obtenu est donc :
ordinateur d'Alan → R8 → R7 → R6 → ordinateur d'Ada.

Le chemin emprunté est : R8 → R5 → R6 → R1 → R3.
Il a un coût de 3 * 10 + 1 = 31.
Tous les autres chemins passent par une connexions Ethernet de coût 100.

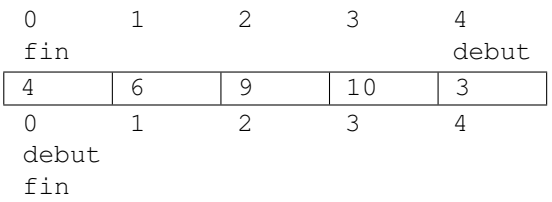
**Exercice 3 (architecture matérielle, langage assembleur, structures linéaires de données, programmation orienté objet)**

1. Il s'agit de von Neumann, pour le modèle de von Neumann
2. 1) Mémoire RAM, 2) Processeur, 3) Unité de contrôle, 4) Unité arithmétique et logique
3. A chaque extinction de l'ordinateur, la mémoire RAM est perdue, c'est pour cela qu'on dit qu'elle est volatile.
4. De la plus rapide à la plus lente pour l'accès aux données, les mémoires sont : Registre, Mémoire vive, Mémoire cache, Disque dur
5. Il s'agit de l'instruction : `sub r2, r5, r4`
6. (erreur dans le sujet, il aurait du y avoir écrit `addi r1 r2 0`) Cela place dans le registre r1 la somme de la valeur du registre r2 et de la valeur 0. Donc cela place dans le registre r1 la valeur du registre r2.
7. On peut écrire :
```
addi r3 r1 0
addi r1 r2 0
addi r2 r3 0
```
8. A = 0, B = 1, C = 1, D = 1
9. Le circuit est un additionneur binaire, il fait la somme de E1 + E2 + Re et donne le résultat $(RsS)_2$ où Rs est le bit de retenue et S est le bit de même niveau que les entrées E1 et E2.

**Partie B**

10. Dans une file, le premier arrivé est le premier servi, en anglais on dit FIFO pour First in First out.
11. La valeur de `contenu` de l'objet `memoire` à l'issue de la commande est `[None, None, None, None, None]`
12. On obtient les états successifs suivants :

| | | | | |
|---|---|---|---|---|
| 4 | 6 | 9 | None | None |
| 0 debut | 1 | 2 | 3 fin | 4 |
| 4 | 6 | 9 | None | None |
| 0 | 1 debut | 2 | 3 fin | 4 |
| 4 | 6 | 9 | 10 | None |
| 0 | 1 debut | 2 | 3 | 4 fin |
| 4 | 6 | 9 | 10 | None |
| 0 | 1 | 2 debut | 3 | 4 fin |
| 4 | 6 | 9 | 10 | None |
| 0 | 1 | 2 | 3 debut | 4 fin |
| 4 | 6 | 9 | 10 | 3 |
| 0 fin | 1 | 2 | 3 debut | 4 |
| 4 | 6 | 9 | 10 | 3 |




13. L'attribut `contenu` a pour valeur `['o', 's', 'i', 'n', 'f']`.
14. On obtient le code suivant :

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

15. On obtient le code suivant :

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