Proposition de correction

Exercice 1

Partie A

Q1

>>> 8
>>> [8, 7, 18, 16, 12, 9, 17, 3]

Q2

```
for i in range(2,5):
    print(notes[i])
```

Partie B

Q1

```
def tri_insertion(liste):
    """ trie par insertion la liste en paramètre """
    for indice_courant in range(1, len(liste)):
        element_a_inserer = liste[indice_courant]
        i = indice_courant - 1
        while i >= 0 and liste[i] > element_a_inserer:
            liste[i+1] = liste[i]
            i = i - 1
        liste[i + 1] = element_a_inserer
```

Q2

>>> [7, 8, 18, 14, 12, 9, 17, 3]

Q3

>>> [7, 8, 14, 18, 12, 9, 17, 3]

Partie C

Q1

récursif : à l'étape (3) l'algorithme de tri fusion s'appelle lui-même.

Q2

1. Comparer les cartes du haut des 2 tas.
2. Placer la carte de valeur plus faible dans la main.
3. Recommencer l'étape 1 jusqu'à épuisement des tas.

Q3

```
from math import floor
```

```
def tri_fusion (liste, i_debut, i_fin):
    """ trie par fusion la liste en paramètre depuis i_debut jusqu'à i_fin """
    if i_debut < i_fin:
        i_partage = floor((i_debut + i_fin) / 2)      # milieu pour diviser le tableau en deux moitiés
        tri_fusion(liste, i_debut, i_partage)         # Appel tri_fusion pour 1ère moitié du tableau
        tri_fusion(liste, i_partage + 1, i_fin)       # Appel tri_fusion pour 2ème moitié du tableau
        fusionner(liste, i_debut, i_fin, i_partage) # Fusion des deux moitiés triées
```

Q4

permet d'importer la fonction floor() du module math utilisée à la ligne 7

partie D

Q1

tri par fusion : à chaque étape, le tri se fait par fusion de 2 tas déjà triés

Q2

- tri par insertion : $O(n^2)$
- tri par fusion : $O(n.ln_2 n)$

Q3

- Le tri par insertion utilise 2 boucles imbriquées, soit dans le pire des cas ~ 1/2n(n+1) opérations.
- Le tri par fusion, divise la liste à trier par 2 pour chaque itérations, soit ~ $ln_2 n$ opérations.

Exercice 2

Partie A

Q1

- Clients : IdClient
- Articles : IdArticle

Q2

- Email : VARCHAR(50), chaîne de 50 caractères
- Quantity : INT, entier signé

Q3

```
CREATE TABLE Commandes (
        IdCmd                   INT PRIMARY KEY,
        IdClient                INT,
        Date                    DATE,
        AdresseLivraison        VARCHAR(90),
        PaiementValide          BOOLEAN,
        LivraisonFaite          BOOLEAN,
        FOREIGN KEY(IdClient) REFERENCES Clients (IdClient)
        );
```

Partie B

Q1

La méthode POST fait passer les informations à envoyer dans le corps de la requête HTTP et ne figurent pas dans l'adresse URL du serveur, contrairement à la méthode GET.

Q2

Le protocole HTTPS est chiffré de bout en bout ce qui est indispensable pour assurer la confidentialité des transactions bancaires.

Q3

Afin d'assurer la cohérence des données en vue de leur traitement ultérieur. Ex : envoi d'un mailing.

Partie C

Q1

SELECT IdArticle, Libelle FROM Articles WHERE PrixEnCentimes < 1500

Q2

La requête sélectionne tous les identifiants et les mèls des clients, ainsi que les identifiants et les adresses de livraison de leurs commandes pour des paiements qui n'ont pas été validés.

Q3

SELECT Articles.Libelle

FROM Articles, ArticlesCommande, Commandes

WHERE Commandes.IdCmd = 1345 AND ArticlesCommande.IdCmd = Commandes.IdCmd AND Articles.IDArticle = ArticlesCommande.IdArticle

Q4

INSERT INTO Articles VALUES(NULL, 'Imperméable', 'Cet imperméable se replie en forme de pochette.', 999)

Partie D

Q1

- Relation Articles : ajout attribut stock INT
- Relations Client : ajout attribut AdresseLivraison VARCHAR(90)

Q2

Il ne faut pas décrémenter stock de 1 mais de la quantité commandée : Stock ← Stock – Quantité

Exercice 3

Partie A

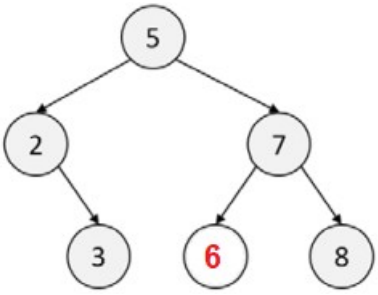
Q1

- Racine : 5
- Fils gauche : 2
- Fils droit : 7

Q2

Noeuds : [5, 2]

Q3



Partie B

Q1

__init__ initialise les attributs de l'objet lors de l'instanciation de la classe.

Q2

L'élément à insérer est ignoré.

Q3

arbre = ABR(5)
arbre.insererElement(2)
arbre.insererElement(3)
arbre.insererElement(7)
arbre.insererElement(8)

Partie C

Q1

Parcours infixe

Q2

La complexité est de l'ordre du tri par fusion ~ $O(n\ln_2 n)$

Alors que les complexités des tris par sélection et insertion sont ~ $O(n^2)$

Exercice 4

Partie A

Q1

Le réseau fonctionne sur le principe de la commutation de paquets. TCP découpe le message en paquets (datagrammes). Ces paquets sont routés par IP grâce aux en-têtes qui encapsulent le message.

Q2a

- Le sommets représentent le nom des routeurs.
- Les arêtes représentent les connexions entre les routeurs.

Q2b

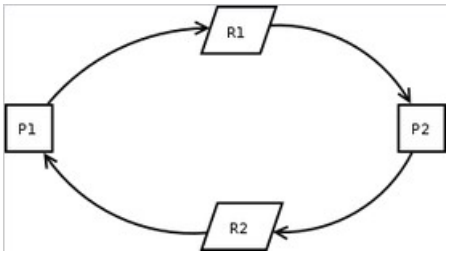
RIP utilise le nombre de sauts entre les routeurs.

Partie B

Q1

L'interblocage se produit lorsque des processus concurrents s'attendent mutuellement.

Exemple d'interblocage : le processus *P1* utilise la ressource *R2* qui est attendue par le processus *P2* qui utilise la ressource *R1*, attendue par *P1*.

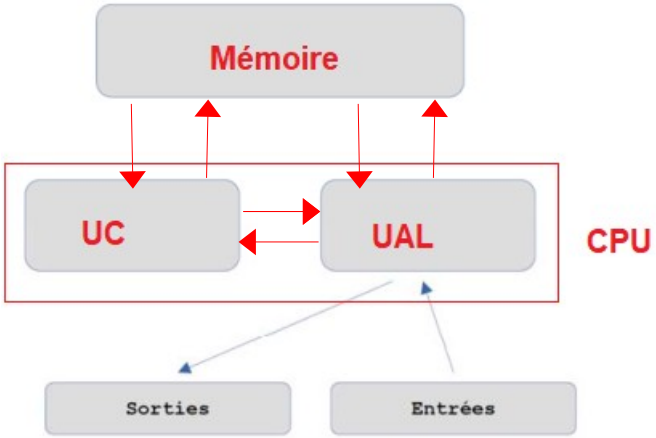


Q2

- Utilisation de mutex.
- Algorithme du Banquier.

Partie C

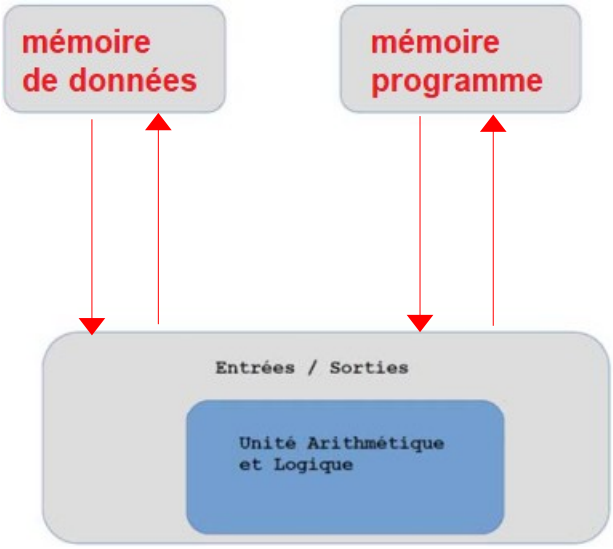
Q1



Q2

- IP : UC. Registre qui contient l'adresse mémoire de l'instruction en cours d'exécution ou prochainement exécutée. Une fois l'instruction chargée, il est automatiquement incrémenté pour pointer l'instruction suivante.
- IR : UC. Registre qui contient l'instruction en cours d'exécution ou de décodage.

Q3



Q4

- Mémoire morte : mémoire non volatile dont le contenu est fixé lors de leur fabrication, qui peut être lue plusieurs fois et qui n'est pas prévue pour être modifiée.
- Mémoire vive : mémoire volatile dans laquelle des données peuvent être écrites, lues et modifiées.

Un microcontrôleur exécute un programme dès qu'il est mis sous tension. Ce programme ne doit pas être modifié par un simple utilisateur.

Partie D

Q1

Chaque microprocesseur peut effectuer une tâche spécialiser pour augmenter la rapidité du traitement.

Q2

La vitesse des différents composants (CPU, RAM, circuits d'entrées/sorties, ...) évolue en fonction de la technologie. Cette évolution n'est pas la même pour tous les composants.

Q3

La miniaturisation permet d'avoir des fonctions avancées de taille réduite, facile à intégrer physiquement dans des systèmes dits embarqués.

Q4

La puissance de calcul est généralement plus faible qu'avec une architecture classique.

Exercice 5

Partie A

Q1

- Relation Films : IdFilm
- Relation Abonnés : IdAbonne

Q2

- IdFilm : entier non signé
- Description : chaîne de 150 caractères

Q3

- clef primaire : IdCpt
- clef étrangère : IdAbonne

Q4

Il faut créer une Relation Acteurs dont le schéma relationnel est le suivant :

Acteurs(IdActeur, Nom, Prenom)

- La clef primaire est souligné.
- Les attributs Nom et Prenom doivent être de type VARCHAR(20)
- Il faut définir une clef unique sur le couple (Nom, Prenom) pour éviter les doublons.

Et une relation ActeursPrincipaux dont le schéma relationnel est le suivant :

ActeursPrincipaux(#IdActeur, #IdFilm)

- Les clef étrangères commencent par # et référencent respectivement les relations Acteurs et Films.

Q5

- Il faut rajouter un Attribut DateNaissance de type DATE à la relation Clients.
- Et ajouter un attribut AgeMinimum de type INT à la relation Films.

Partie B

Q1

SELECT IdCpt, Pseudo FROM ComptesAbonnes WHERE IdAbonne = 237

Q2

Calcule la moyenne du nombre d'étoiles attribué au film dont l'identifiant est 1542

Q3

Sélectionne, par ordre décroissant en nombre d'étoiles, les identifiants, les titres des films et les nombres d'étoiles attribués par le compte abonné n° 508

Q4

UPDATE ComptesAbonnes SET pseudo = 'Champion' WHERE IdAbonne = 508

Partie C

Q1

La fonction calcule, dans une liste de films non vide, la moyenne des écarts des notes attribuées à chaque film par deux comptes d'abonnés.

Q2

```
def conseilsFilms(IdCpt : int) -> list:
    conseils = []
```

```
    liste_films      = podiumCompte(IdCpt)
    liste_spectateurs = spectateurs(liste_films)
    for spectateur in liste_spectateurs:
        if distance(IdCpt, spectateur, liste_films) < 10:
            film_preferes = podiumCompte(spectateur)
            for i in range(3):
                if i < len(film_preferes):
                    conseils.append(film_preferes[i])

    return conseils
```