Centres étrangers 2023 Jour 1

Exercice 1

1. Schéma relationnel de la relation `description` (la clé primaire est soulignée et la clé étrangère est précédée du symbole #):

   description(id_description : INT, resume : TEXT, duree : INT, #id_emission : INT)

2. a. La requête affiche le résultat suivant :

theme	annee
Le système d'enseignement supérieur français est-il juste et efficace ?	2022
Trois innovations pour la croissance future (1/3) : La révolution blockchain	2021

   b. Requête permettant d'afficher les thèmes des podcasts de l'année 2019 :

```
SELECT theme
FROM podcast
WHERE annee = 2019
```

   c. Requête permettant d'afficher la liste des thèmes et des années de diffusion des podcasts dans l'ordre chronologique des années :

```
SELECT theme, annee
FROM podcast
ORDER BY annee
```

3. a. La requête proposée affiche la liste de tous les thèmes de la relation `podcast` **sans répétition**.

   b. Requête SQL supprimant la ligne contenant l'`id_podcast = 40` de la relation `podcast` :

```
DELETE FROM podcast
WHERE id_podcast = 40
```

4. a. Requête SQL permettant de changer le nom de l'animateur de l'émission "Le Temps de débat" en "Emmanuel L".

```
UPDATE emission
SET animateur = "Emmanuel L"
WHERE nom = "Le Temps de débat"
```

   b. Requête SQL permettant d'ajouter l'émission "Hashtag" sur la radio "France inter" avec "Mathieu V.", avec un `id_emission` égal à 12850.

```
INSERT INTO emission (id_emission, nom, radio, animateur)
VALUES (12850, "Hashtag", "France inter", "Mathieu V.")
```

5. Requête permettant de lister les thèmes, le nom des émissions et le résumé des podcasts pour lesquels la durée est strictement inférieure à 5 minutes.

```
SELECT podcast.theme, emission.nom, description.resume
FROM podcast
JOIN emission ON podcast.id_emission = emission.id_emission
JOIN description ON emission.id_emission = description.id_emission
WHERE description.duree < 5
```

> **!** Important
>
> La base de donnée telle que définie dans l'énoncé n'est bien construite. Chaque description est en-effet reliée à une émission unique à travers la clé étrangère `id_emission`, mais pas au podcast correspondant. Comme il existe plusieurs podcasts pour une émission, il n'y a pas moyen de savoir à quel podcast correspond quelle description.

Exercice 2

1. a. On convertit les entiers 2 et 13 en binaire sur 8 bits :

2	13
00000010	00001101

L'adresse IP 164.178.2.13 est donc représentée par la chaîne de caractères `10100100.10110010.00000010.0`

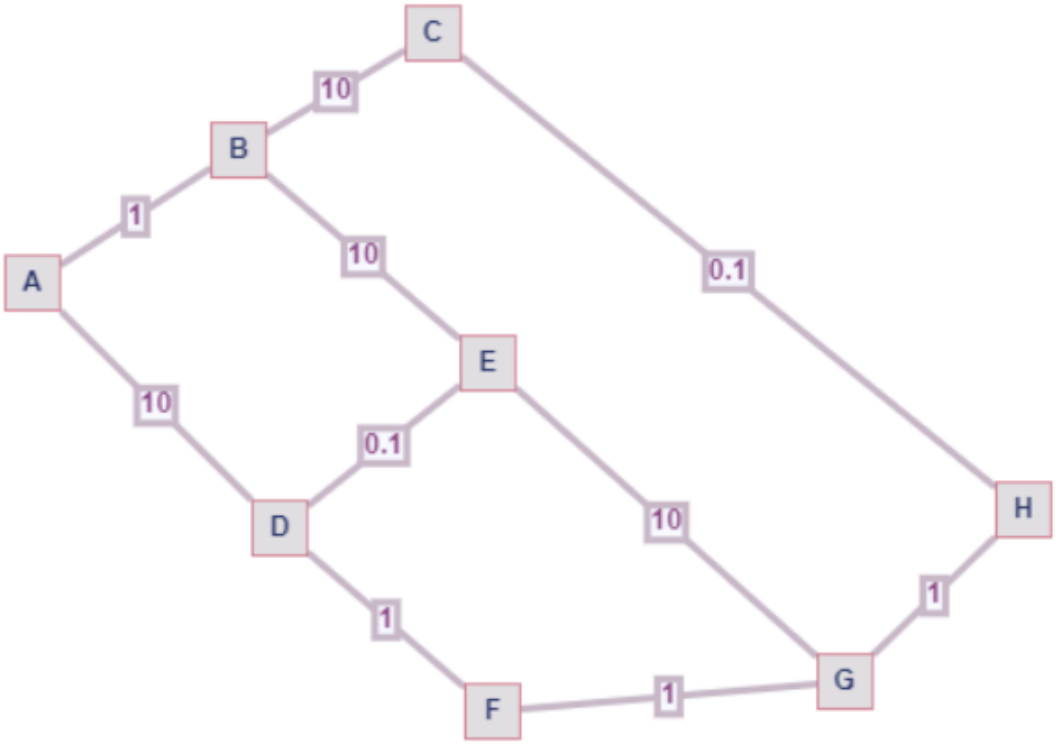
b. L'adresse IP indique que les 24 premiers bits sont réservés à l'identifiant du réseau et les 8 derniers bits sont réservés à l'identifiant de l'hôte. La machine appartient donc au réseau dont l'adresse est 164.178.2.0

2. Pour un paquet émis par A à destination de G, les chemins optimaux en suivant le protocole RIP sont ceux qui minimisent le nombre se sauts :

Chemin	Nombre de sauts
A -> B -> C -> H -> G	4
A -> B -> E -> G	3
A -> D -> E -> G	3
A -> D -> F -> G	3

Les chemins optimaux sont donc A -> B -> E -> G, A -> D -> E -> G et A -> D -> F -> G.

3. a. Le coût d'une liaison Ethernet est $\frac{10^9}{10^8} = 10$, celui d'une liaison Fast-Ethernet est $\frac{10^9}{10^9} = 1$ et celui d'une liaison fibre est $\frac{10^9}{10^{10}} = 0.1$



b. Nous pouvons déterminer le chemin de parcours en utilisant l'algorithme de Dijkstra.

A	B	C	D	E	F	G	H	choix
0-A	-	-	-	-	-	-	-	A(0)
X	1-A	-	10-A	-	-	-	-	B(1)
X	X	11-B	10-A	11-B	-	-	-	D(10)
X	X	11-B	X	10.1-D	11-D	-	-	E(10.1)
X	X	11-B	X	X	11-D	20.1-E	-	C(11) ou F(11)
X	X	X	X	X	X	12-F	11.1-C	H(11.1)
X	X	X	X	X	X	12-F	X	G(12)

Le chemin optimal est donc A -> D -> F -> G de coût total 12.

c. Le routeur F est en panne. Nous appliquons l'algorithme de Dijkstra en supprimant le routeur F

A	B	C	D	E	G	H	choix
0-A	-	-	-	-	-	-	A(0)
X	1-A	-	10-A	-	-	-	B(1)
X	X	11-B	10-A	11-B	-	-	D(10)
X	X	11-B	X	10.1-D	-	-	E(10.1)
X	X	11-B	X	X	20.1-E	-	C(11)
X	X	X	X	X	20.1-E	11.1-C	H(11.1)
X	X	X	X	X	12.1-H	X	G(12.1)

Le chemin optimal est donc A -> B -> C -> H -> G de coût total 12.1.

Exercice 3

1. Fonction permettant l'ajout d'une couleur aléatoire dans la file `f` :

```
def ajout(f):
    couleurs = ("bleu", "rouge", "jaune", "vert")
    indice = randint(0, 3)
    enfiler(f, couleurs[indice])
    return f
```

2. Fonction permettant de vider la séquence `f` :

```
def vider(f):
    while not est_vide(f):
        defiler(f)
```

3. Fonction `affich_seq` complétée :

```
def affich_seq(sequence):
    stock = creer_file_vide()
    ajout(sequence)
    while not est_vide(sequence):
        c = defiler(sequence)
        affichage(c)
        time.sleep(0.5)
        enfiler(stock, c)
    while not est_vide(stock):
        enfiler(sequence, defiler(stock))
```

4. a. Fonction `tour_de_jeu` complétée :

```
def tour_de_jeu(sequence):
    affich_seq(sequence) # zone A
    stock = creer_file_vide()
    while not est_vide(sequence):
        c_joueur = saisie_joueur()
        c_seq = defiler(sequence) # zone B
        if c_joueur == c_seq:
            enfiler(stock, c_seq) # zone C
        else:
            vider(sequence) # zone D
    while not est_vide(stock): # zone E
        enfiler(sequence, defiler(stock)) # zone F
```

b. Fonction modifiée :

```
def tour_de_jeu(sequence):
    affich_seq(sequence) # zone A
    stock = creer_file_vide()
    gagne = True
    while not est_vide(sequence):
        c_joueur = saisie_joueur()
        c_seq = defiler(sequence) # zone B
```

```
        if c_joueur == c_seq:
            enfiler(stock, c_seq) # zone C
        else:
            vider(sequence) # zone D
            gagne = False
    while not est_vide(stock): # zone E
        enfiler(sequence, defiler(stock)) # zone F
    if gagne:
        tour_de_jeu(sequence)
    else:
        vider(sequence)
        ajout(sequence)
        tour_de_jeu(sequence)
```