DEVOIR COMMUN

NUMÉRIQUE ET SCIENCES INFORMATIQUES

Durée de l'épreuve : 3 heures 30
L'usage de la calculatrice n'est pas autorisé.
Dès que ce sujet vous est remis, assurez-vous qu'il est complet.
Ce sujet comporte 8 pages numérotées de 1 / 8 à 8 / 8.
Le sujet est composé de trois exercices indépendants.

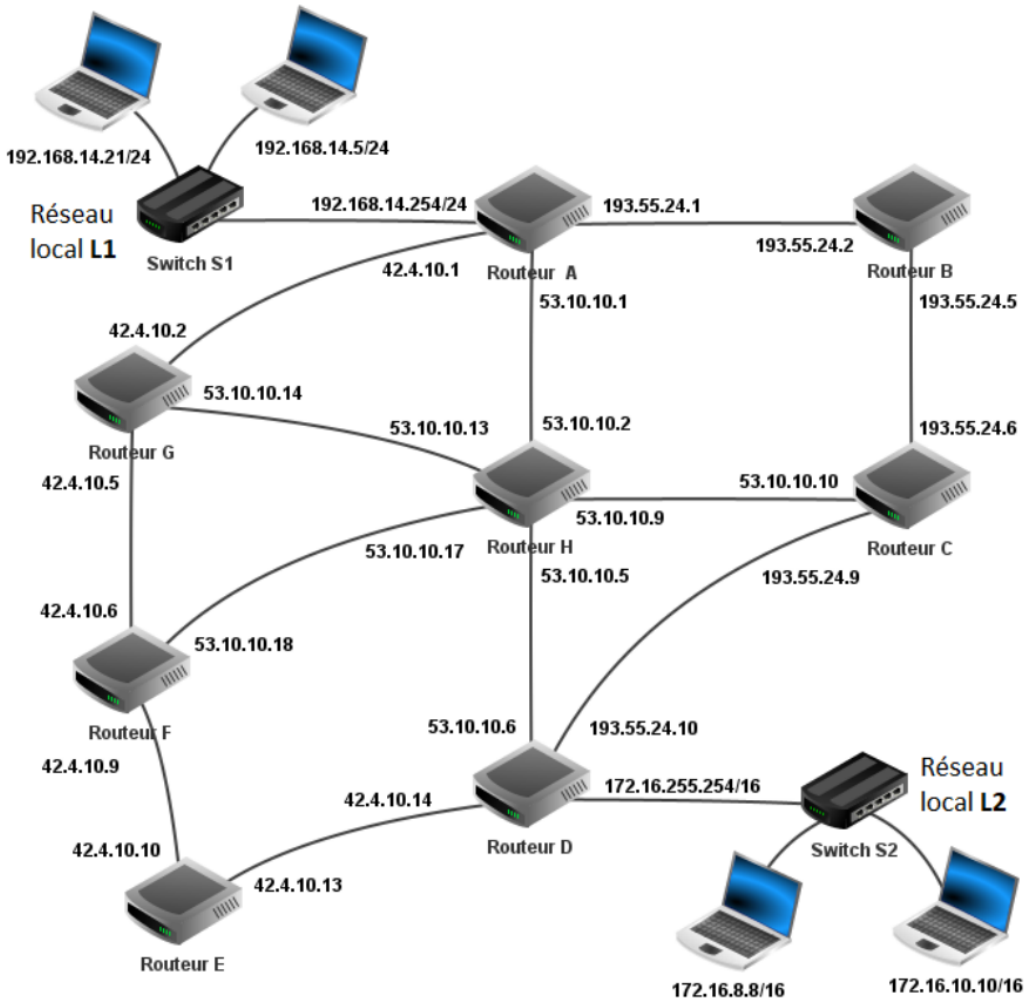
Exercice 1 6 points

i Cet exercice traite d'architecture réseau et de routage.

On liste ci-dessous les puissances de 2.

$2^0 = 1$	$2^9 = 512$	$2^{18} = 262144$	$2^{27} = 134217728$
$2^1 = 2$	$2^{10} = 1024$	$2^{19} = 524288$	$2^{28} = 268435456$
$2^2 = 4$	$2^{11} = 2048$	$2^{20} = 1048576$	$2^{29} = 536870912$
$2^3 = 8$	$2^{12} = 4096$	$2^{21} = 2097152$	$2^{30} = 1073741824$
$2^4 = 16$	$2^{13} = 8192$	$2^{22} = 4194304$	$2^{31} = 2147483648$
$2^5 = 32$	$2^{14} = 16384$	$2^{23} = 8388608$	$2^{32} = 4294967296$
$2^6 = 64$	$2^{15} = 32768$	$2^{24} = 16777216$	
$2^7 = 128$	$2^{16} = 65536$	$2^{25} = 33554432$	
$2^8 = 256$	$2^{17} = 131072$	$2^{26} = 67108864$	

On considère le réseau représenté ci-dessous :



• Partie A : Adresses IP

1. Donner le masque de sous réseau du réseau L2 est /16 en notation CIDR. Donner la notation décimale pointée de ce masque. .

2. Concernant le réseau local L2 :
   (a) Donner l'adresse du réseau.
   (b) Donner l'adresse de diffusion.
   (c) Donner le nombre maximum de machines pouvant être connectées à ce réseau.

• Partie B : Protocoles de routage

On donne ci-dessous des extraits des tables de routage des routeurs :

Routeur	Réseau destinataire	Passerelle	Interface
A	L2	53.10.10.2	53.10.10.1
B	L2	193.55.24.6	193.55.24.5
C	L2	193.55.24.10	193.55.24.9
D	L2	Connecté	172.16.255.254
E	L2	42.4.10.14	42.4.10.13
F	L2	42.4.10.10	42.4.10.9
G	L2	53.10.10.13	53.10.10.14
H	L2	53.10.10.6	53.10.10.5

3. À l'aide des extraits des tables de routage ci-dessus, donner un chemin (c'est à-dire nommer les routeurs traversés) suivi par un message envoyé du réseau L1 vers le réseau L2.

La liaison entre les routeurs H et D est rompue :

4. Sachant que le protocole de routage RIP est utilisé (distance en nombre de sauts), donner les nouveaux chemins que pourra suivre un message allant de L1 vers L2.
5. Choisir un des chemins de la question précédente. Donner les routeurs dont la règle de routage à destination de L2 est obligatoirement modifiée. Après avoir examiné tous les routeurs, écrire sur votre copie les règles de routage modifiées en conséquence.

La liaison entre les routeurs H et D est rétablie.

Pour tenir compte du débit des liaisons, on décide d'utiliser le protocole OSPF (distance liée au coût des liaisons) pour effectuer le routage.

Le coût d'une liaison est donné ici par la formule :

$$\text{cout} = \frac{10^8}{BP}$$

où BP est la bande passante de la connexion en bit par seconde.

Les valeurs des bandes passantes de chaque liaison entre les routeurs sont données ci-dessous :

Liaison	Bande passante
A-B	1 Gbit/s
A-H	1 Gbit/s
A-G	1 Gbit/s
B-C	1 Gbit/s
C-H	100 Mbit/s
C-D	1 Gbit/s

Liaison	Bande passante
D-H	100 Mbit/s
D-E	10 Gbit/s
E-F	10 Gbit/s
F-H	1 Gbit/s
F-G	10 Gbit/s
G-H	1 Gbit/s

1 Kbit/s = 10^3 bit/s 1 Mbit/s = 10^6 bit/s 1 Gbit/s = 10^9 bit/s

6. Calculer le coût des liaisons pour les 3 valeurs de bande passante qui apparaissent dans le tableau ci-dessus.
7. Déterminer alors le chemin que suivra un message allant de L1 vers L2 et donner son coût.

Exercice 2 6 points

i Cet exercice traite de la programmation orientée objet.
On rappelle quelques fonctions et méthodes des tableaux (le type list en Python) qui pourront être utilisées dans cet exercice :

- `len(tab)` : renvoie le nombre d'éléments du tableau `tab` ;
- `tab.append(elt)` : ajoute l'élément elt en fin de tableau `tab` ;
- `tab.remove(elt)` : enlève la première occurrence de `elt` de `tab` si `elt` est dans t`tab` . Provoque une erreur sinon.

Une célèbre plateforme de livraison vous engage pour concevoir une partie de son logiciel. Votre mission est de modéliser un système de gestion des commandes à l'aide de la programmation orientée objet.

Ce système doit permettre de créer de créer des commandes, d'y ajouter ou retirer des titres musicaux, et d'afficher des informations utiles comme le prix total de la commande.

La classe `Plat`

Pour cela, on a commencé à créer la classe `Plat`.

```
1 class Plat:
2     def __init__(self, n, t, p, tp):
3         self.nom = n
4         self.type = t
5         self.prix  = p
6         self.temps_preparation = tp
```

1. Citer les attributs de classe `Plat`
2. Quelle instruction permet d'instancier un objet `pizza4` qui est à pour nom "Pizza 4 fromages", qui coûte 11,50 € et qui est de type "`plat principal`" qui nécessite 10 minutes de préparation.
3. Écrire la méthode `infos_plat` de la classe `Plat` qui `affiche` les informations sur un plat.

   Par exemple, l'instruction `pizza4.infos_plat()` devra afficher :

   `Pizza 4 fromages :=> Type : plat principal - Prix : 11.50€`

La classe `Commande`

4. Créez une classe Commande avec les attributs suivants :
   - `nom` : (`str`) le nom de la commande.
   - `client` : (`str`) le nom du client.

— `plats` : (`list`) une tableau (vide au départ) qui contiendra les objets `Plats`.

5. Créer une méthode `nb_plats` pour la classe `Commande` qui retourne le nombre de plats de la commande.
6. Créer une méthode `ajouter_plat` pour la classe `Commande` qui prend en paramètre `pt` un objet de type `Plats` à l'attribut `plats`.
7. `afficher_commande` : affiche toutes les informations des plats de la commande en utilisant la méthode `infos_plat` de chaque plat.
8. Créer une méthode `prix_total` pour la classe `Commande` qui renvoie le prix total de la commande.
9. Pour gagner du temps et que la préparation de la commande soit la plus rapide possible, il faut commencer les plats les plus long à préparer.

   Écrire une méthode `a_preparer_en_premier`, qui renvoie le `nom` du plat de la commande le plus long à préparer.
10. Créer une méthode `supprimer_plat` pour la classe `Commande` qui prend en paramètre `nom_de_plat` une chaîne de caractères et qui enlève le plat du nom `nom_de_plat` de l'attribut `plats`.

    Attention la méthode `supprimer_plat` ne doit en aucun cas générer une erreur.

Exercice 3 8 points

i Cet exercice porte sur la programmation Python (dictionnaire), la programmation orientée objet, les bases de données relationnelles et les requêtes SQL.
On pourra utiliser les mots du langage SQL suivants : SELECT, FROM, WHERE, JOIN, INSERT INTO, VALUES, COUNT.

Un étudiant souhaite développer une application permettant de faciliter le covoiturage pour les déplacements du quotidien. Dans cet objectif, il étudie des données extraites de la Base Nationale des Lieux de Covoiturage (BNLC), disponible sur le site data.gouv.fr.
Dans un premier temps (partie A), les données d'une table décrivant des lieux de covoiturage (adresse postale, nombre de places, ...) sont manipulées a l'aide d"un tableau contenant des dictionnaires en langage Python.
Dans un second temps (partie B), une base de données contenant deux tables (les sites de covoiturage et les caractéristiques des communes de France) est exploitée l'aide du langage SQL.

• Partie A : traitement de données en table

Une table est implémentée par un tableau nommé `tab_lieux` contenant des dictionnaires en langage Python.
Chaque dictionnaire correspond un lieu de stationnement pour le covoiturage.
Les clés des dictionnaires, communes a tous les dictionnaires, correspondent aux descripteurs utilisés pour cette table :

— `id_lieu` : identifiant du lieu, la donnée est un entier (chaque lieu possède un identifiant unique) ;

— `ad_lieu` : adresse du lieu, la donnée est une chaîne de caractères ;

— `insee` : code INSEE de la commune ou se trouve le lieu, la donnée est une chaîne de caractères ;

— `nb_places` : nombre de places du lieu, la donnée est un entier ;

— `type` : nature du parking (supermarché, parking municipal, aire de stationnement située en sortie d'autoroute, ...), la donnée est une chaîne de caractères.

On donne en illustration les trois premiers éléments de ce tableau de dictionnaires,

```
tab_lieux = [
    {"id _lieu": 1, "ad_lieu": "Place De La Fontaine", "insee": "1024",
    ↪ "nb_places": 5, "type": "Supermarché"},
    {"id_lieu": 2, "ad_lieu": "La Boisse", "insee": "1049", "nb _places": 100,
    ↪ "type": "Parking municipal"},
    {"id_lieu": 3, "ad lieu": "Chateau-Gaillard", "insee": "1089", "nb_places": 15,
    ↪ "type": "Sortie autoroute"},
    ]
```

1. Accès aux informations du tableau :
   (a) Donner, sans justifier, la valeur a laquelle on accède avec l'instruction `tab_lieux[1]["insee"]`.
   (b) Écrire l'instruction qui permet d'obtenir la valeur `"Chateau-Gaillard"`.
2. On propose trois blocs d'instructions pour parcourir le tableau et afficher le nombre de places des lieux de covoiturage. Parmi ces trois propositions, deux seulement sont correctes. Indiquer, sans justifier, les deux propositions correctes.

Proposition 1 :

```
for i in range(len(tab_lieux)):
    print (dico["nb_places"])
```

Proposition 2 :

```
for i in range(len(tab_lieux)):
    print (tab_lieux[i]["nb_places"])
```

Proposition 3 :

```
for dico in tab_lieux:
    print(dico["nb_places"])
```

3. On dispose de la fonction ci-dessous :

```
def mystere(table, n, nom_type):
    """
    Entrée : un tableau de dictionnaires, un entier, une chaîne de caractères
    Sortie : un entier
    """
    k = 0
    for dico in table:
        if dico["nb_places"] >= n and dico["type"] == nom_type:
            k=k+1
    return k
```

Décrire, dans le contexte de l'exercice, ce que renvoie cette fonction, lors de l'appel : `mystere(tab_lieux, 100, "Sortie autoroute")`

4. La fonction, dont le code est proposé ci-après, permet de renvoyer le code INSEE du lieu de covoiturage possédant le plus grand nombre de places. Recopier et compléter ce code

```
def insee_max_places(table):
    """
    Entrée : un tableau de dictionnaires
    Sortie : une chaîne de caractères (le code INSEE du lieu
    possédant le plus grand nombre de places)
    """
    maxi = ...
    code_insee = ...
    for dico in table:
        if ....
            maxi = ...
            code_insee = ...
    return code_insee
```

5. La fonction dont les spécifications sont données ci-après, prend en paramètres un tableau de lieux de covoiturage au format tableau contenant des dictionnaires et le nom d'un type de lieu de covoiturage.

   Recopier et compléter le code de cette fonction. Dans le code de la fonction, les trois points (. . .) peuvent correspondre a une ou plusieurs lignes de programme.

```
def moyenne_par_type(tab, nom_type):
    """
    Entrée : un tableau de dictionnaires et une chaine de
    caractéres (type de lieu)
    Cette fonction renvoie, parmi les lieux de covoiturage
    dont le type est nom_type, la moyenne du nombre de places
    pour le type choisi.
    Sortie : un flottant
    """
    ......
    return moyenne
```

   Exemple, avec la table `tab1` ci-après, contenant uniquement trois enregistrements :

```
tabl = [
    {"id_lieu":1, "ad_lieu":"Place Du marché", "insee":"1032",
    "nb_place":10, "type":"Supermarché"}
    {"id_lieu":2, "ad_lieu":"La Pesse", "insee": "1058",
    "nb_places":100, "type":"Parking municipal"},
    {"id_lieu":3, "ad_lieu":"Le Pautet", "insee":"1075",
    "nb_places":20, "type":"Supermarché"}
    ]
```

   `moyenne_par_type (tab1, "Supermarché")` vaut 15.

• Partie B : base de données

Afin de pouvoir gérer un site de covoiturage, on utilise une base de données contenant les relations `Lieux` et `Communes`.

On donne un **extrait des deux premières lignes** de ces relations :

Lieux				
id_lieu	ad_lieu	code_insee	nb_places	type
1	”Place De La Fontaine”	”01001”	5	”Supermarché”
2	”La Boisse”	”01049”	100	”Parking municipal”

Communes					
code_insee	nom	departement	Region	latitude	longitude
”01001”	”L’ABERGEMENT	”AIN”	”RHONE-ALPES”	46.1534	4.9261
”01002”	”VAREY”	”AIN”	”RHONE-ALPES”	46.0092	5.4280

6. Sachant que le `code_insee` est une clé primaire pour la relation `Communes`. Écrire le schéma relationnel de la base de données.
   Rappel : les clés primaires sont soulignées et les clés étrangères sont précédées d’un #
7. Expliquer pourquoi il a été choisi d’utiliser dans la relation `Communes` le code INSEE plutôt que le nom de la commune comme clé primaire.
8. Écrire une requête SQL permettant d’obtenir l’identifiant `id_lieu` et le code INSEE `code_insee` des lieux de covoiturage de type `"Sortie autoroute"` ordonner par ordre de croissant de leur code INSEE.
9. Écrire une requête SQL permettant de compter le nombre de lieux de covoiturage ayant au moins 100 places.
10. Écrire une requête SQL permettant d’afficher le nom des communes proposant des lieux de covoiturage de type ”Parking municipal”.
11. On souhaite intégrer un nouveau lieu de covoiturage ayant 16 places de type `Parking école` à l’adresse `Place de la mairie` dans la commune ayant pour code INSEE ”38460”. Son identifiant `id_lieux` sera 1058.
12. La commune de code INSEE ”40714” vient d’être intégrée a la commune voisine ”40146”. En conséquence, il faut mettre a jour la base de données. Les lieux liés a la commune de code INSEE ”40714” devront désormais être rattachées a la commune de code INSEE ”40146”.
    Écrire une requête permettant de mettre jour la table `Lieux`.