EXERCICE 3 (8 points) ou (6 points en enlevant la partie A)

Cet exercice porte sur les protocoles réseau, les bases de données relationnelles et les requêtes SQL, l'algorithmique et la programmation en Python.

Cet exercice est composé de 3 parties indépendantes.

Partie A : Le réseau informatique d'un hôpital

Dans un hôpital, le service informatique a créé le réseau ci-dessous :

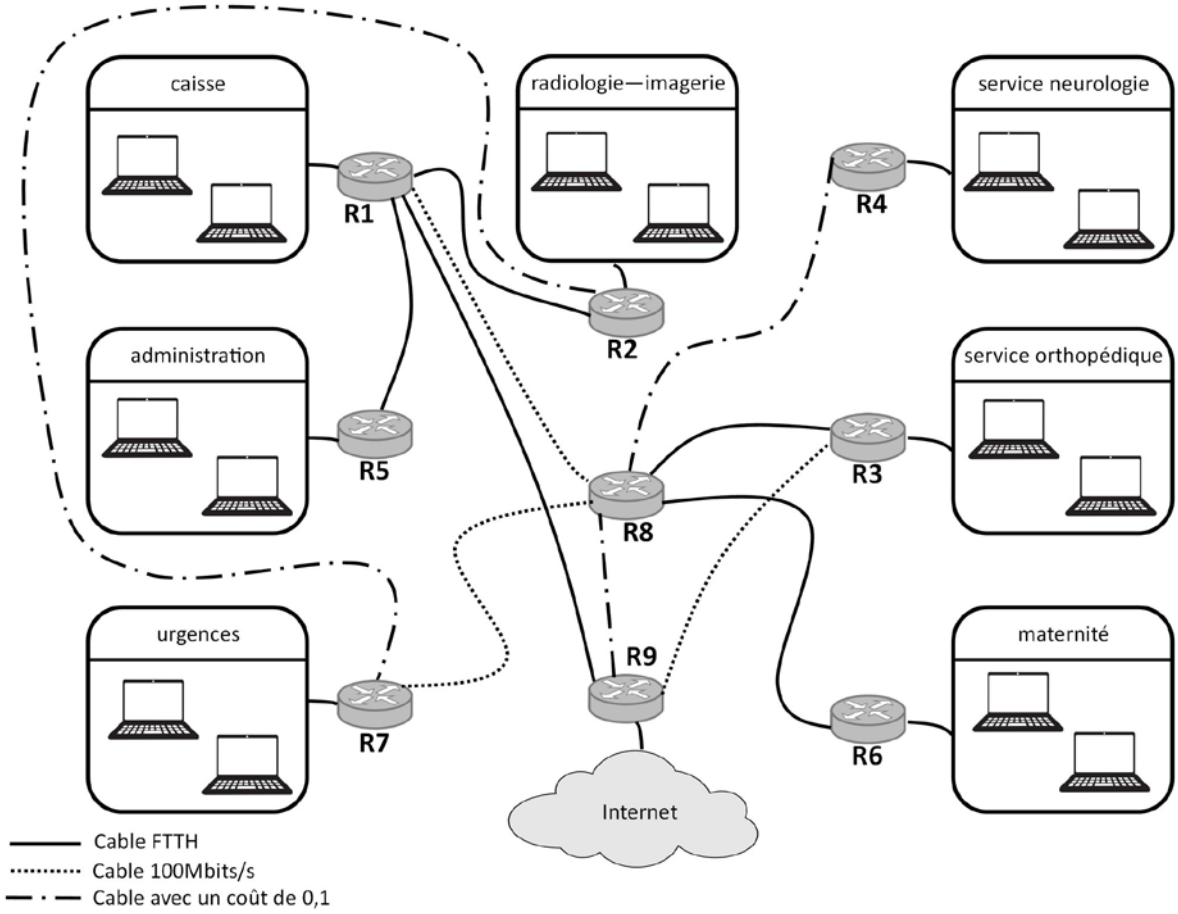


Figure 1. réseau informatique de l'hôpital

Dans un premier temps, on s'intéresse au protocole RIP, qui minimise le nombre de sauts entre routeurs.

1. Donner les chemins possibles d'un paquet de données partant du service de neurologie à destination du service d'imagerie en utilisant le protocole RIP.

2. Recopier et compléter la table des coûts du nœud R2 selon le protocole RIP.

Nœud R2	
Destination	Coût
R1	1
…	…

On s'intéresse maintenant au protocole de routage OSPF. Ce dernier cherche à minimiser la somme des coûts des liaisons entre les routeurs empruntés par un paquet.

Le coût c d'une liaison est donné par :

$$c = \frac{10^8}{d}$$

où d est la bande passante en bit/s de la liaison.

La bande passante des liaisons FTTH (fibre optique : Fiber To The Home) est de 10 Gbit/s et celle des liaisons FastEthernet de 100 Mbit/s.

3. Calculer le coût d'une liaison de communication par la technologie FTTH.

Sur la figure 1, les liaisons sont représentées par un style de trait différents en fonction de leur débit. La légende y est indiquée en bas à gauche.

4. Avec le protocole OSPF, donner le chemin pris par un paquet partant du service de neurologie à destination du service d'imagerie.

Partie B : le dossier médical d'un patient

Lorsqu'un patient se présente dans un hôpital, on lui demande sa carte de sécurité sociale (carte vitale) ainsi qu'une pièce d'identité. En effet, les secrétaires des différents services doivent mettre à jour les données du dossier médical du patient. Ainsi le dossier permet aux médecins de l'hôpital d'avoir accès aux fichiers contenant les divers examens effectués par le patient (les clichés des IRM ou radios, les résultats de ses prises de sang, …).

Pour gérer et partager toutes ces données, le service informatique de l'hôpital a créé une base de données dont le modèle relationnel est donné par le schéma présenté sur la Figure 2. Sur ce schéma, un attribut souligné indique qu'il s'agit d'une clé primaire et un dièse # avant un attribut indique qu'il s'agit d'une clé étrangère.

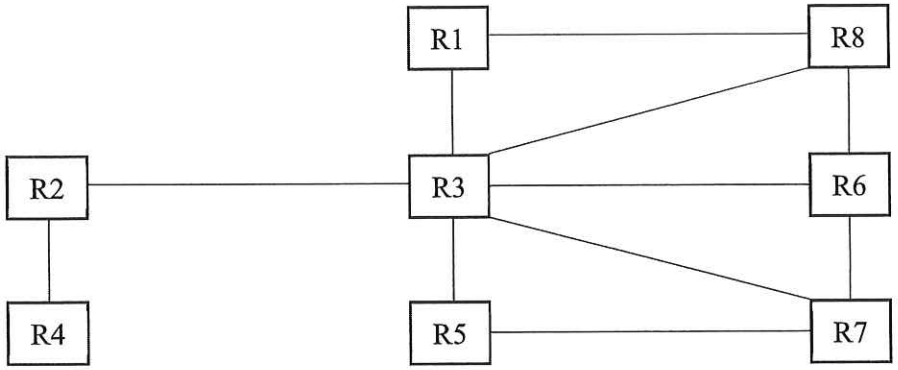
EXERCICE 1 (5 points)

Les deux parties sont indépendantes.

Partie A

La responsable informatique doit gérer le réseau informatique de son entreprise représenté ci-dessous dans lequel R1, R2, R3, R4, R5, R6, R7 et R8 sont des routeurs. Elle décide d'utiliser le protocole RIP pour configurer les tables de routages.

Ce protocole est un protocole à vecteurs de distance. La métrique permettant de décider du meilleur chemin vers un routeur distant est le nombre de sauts.



1. La table de routage de R1 débute ainsi :

Destination	Passerelle	Métrique
R1	R1	0
R2	R3	2
R3	R3	1

Recopier puis compléter la table de routage de R1 selon le protocole RIP.

On donne le constructeur de la classe `Routeur`

```
class Routeur :
    def __init__(self, name) :
        self.nom = name
        self.table_routage = { self : (self, 0) }
        # le routeur est lié à lui-même (self)
```

L'attribut `table_routage` est un dictionnaire dont les clés sont des objets de type `Routeur` et la valeur est le couple `(pasr, m)` où `pasr` est un objet de type `Routeur` et `m` est la métrique selon le protocole RIP entre le routeur de la clé et le routeur `pasr`.

2. La méthode `ajout_destination(self, dest, pasr, m)` de la classe `Routeur`, prend en paramètres un routeur de destination `dest`, un routeur passerelle `pasr` et un entier `m`. Elle ajoute à la table de routage de `self` le routeur `dest` associé à la passerelle `pasr` et à la métrique `m`.

   Par exemple, on crée les routeurs R1, R2 et R3 puis on ajoute les destinations R2 et R3 à la table de routage de R1 à l'aide des commandes suivantes :

```
r1 = Routeur("R1")
r2 = Routeur("R2")
r3 = Routeur("R3")
r1.ajout_destination(r2, r3, 2)
r1.ajout_destination(r3, r3, 1)
```

   On dispose d'une méthode `afficher(self)` qui permet d'obtenir la table de routage du routeur `self`.

   Par exemple, la commande `r1.afficher()` affiche :

```
      ROUTEUR R1 : table de routage
dest.     | pass.    | metrique
R1        | R1       | 0
R2        | R3       | 2
R3        | R3       | 1
```

   a. Proposer les commandes nécessaires permettant d'ajouter les routeurs R4, R5 à la table de routage de R1.

   b. Écrire la méthode `ajout_destination(self, dest, pasr, m)`.

3. Écrire la méthode `voisins(self)` qui renvoie la liste des routeurs directement connectés au routeur `self`.

   Par exemple, `r1.voisins()` renverra la liste `[r3, r8]`, où `r3` et `r8` sont des objets de types `Routeur`.