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 :

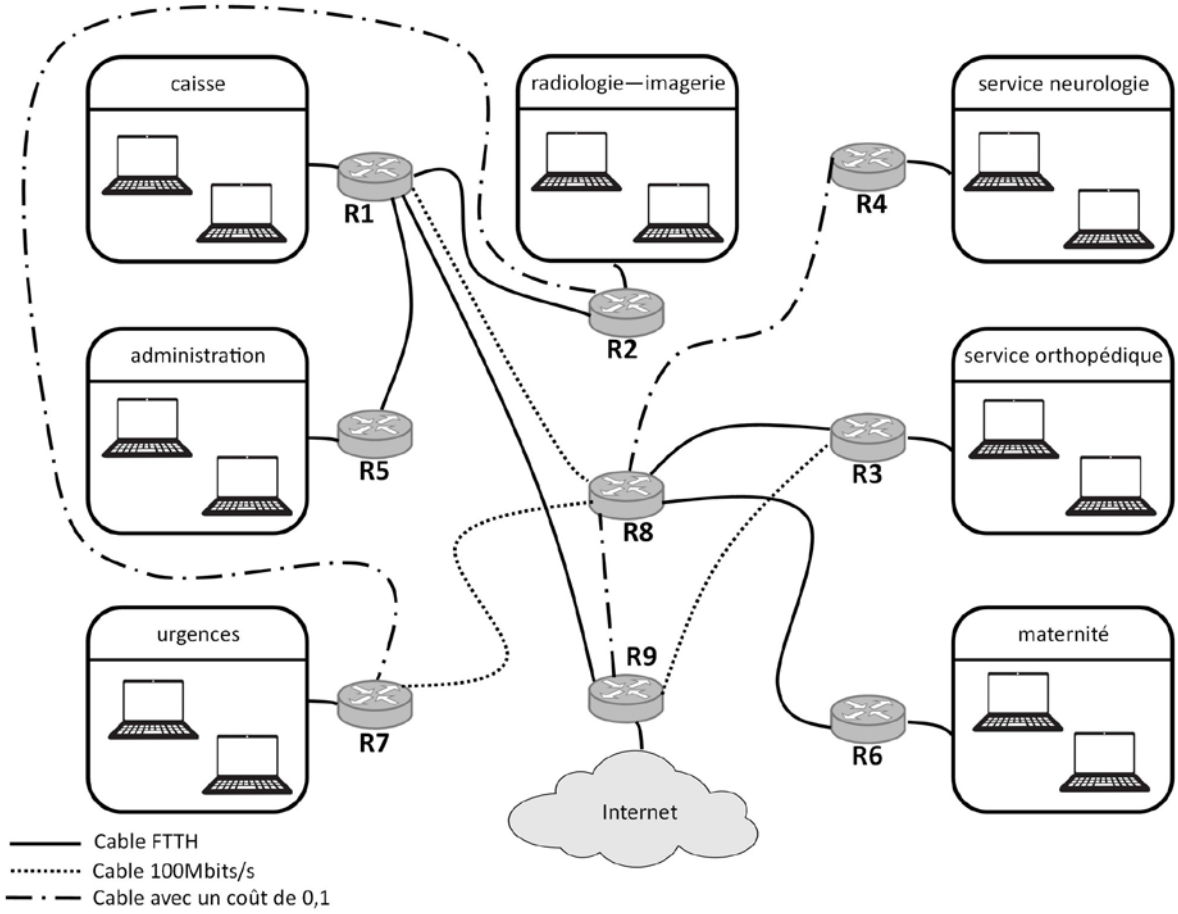


Figure 1. réseau informatique de l'hôpital

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.

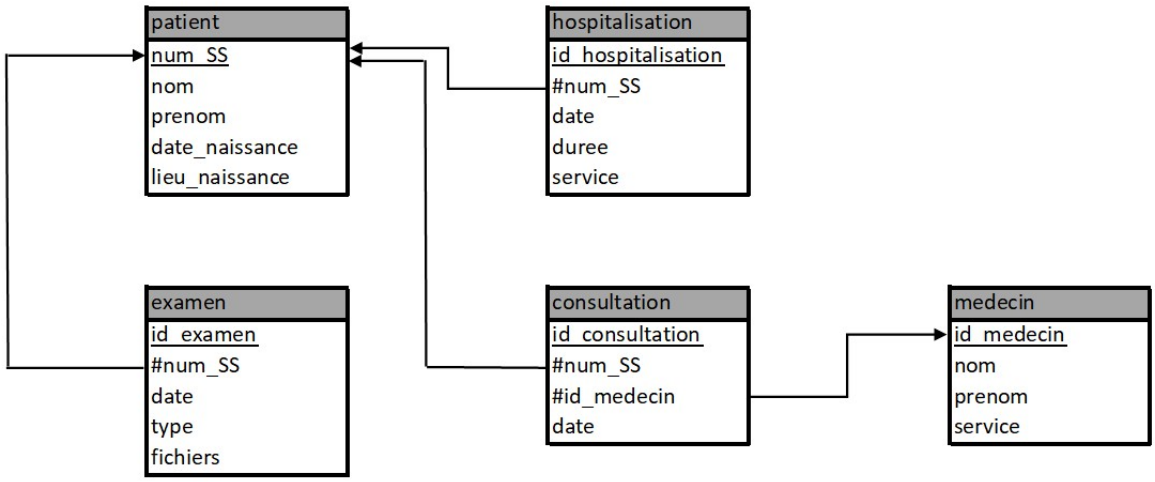


Figure 2. base de données de l'hôpital

Toutes les clés primaires de cette base de données sont de type `INT` (sauf `num_SS` qui est de type `TEXT`), les dates sont de type `DATE` (`JJ/MM/AAAA`) et tous les autres attributs sont de type `TEXT`.

L'énoncé de cet exercice utilise tout ou une partie des mots clefs du langage SQL suivants : `SELECT, DISTINCT, FROM, WHERE, JOIN … ON, UPDATE … SET,DELETE, INSERT INTO … VALUES, LIKE`.

La commande `LIKE 'a%'` permet de rechercher toutes les chaînes de caractères qui commencent par un "a" et `LIKE '%a'` celles qui se terminent par un "a".

Patient				
num_SS	Nom	prenom	date_naissance	lieu_naissance
2 95 07 75 156 189 55	Baujean	Emma	03/07/1995	Paris
1 01 12 69 267 326 21	Tardus	Kylian	16/12/2001	Lyon
2 12 10 33 014 673 82	Delpuis	Sarah	23/10/2012	Bordeaux
1 90 03 37 549 312 43	Montpart	Vincent	30/03/1990	Tours

5. Décrire simplement le résultat obtenu avec la requête SQL ci-dessous :

```
SELECT nom,prenom FROM patient WHERE num_SS LIKE '1%';
```

6. Ecrire une requête SQL permettant d'afficher le numéro de Sécurité Sociale des patients hospitalisés dans le service intitulé `orthopédique` durant l'année 2023.

7. Ecrire une requête SQL permettant d'afficher le type et la date de chaque examen de la patiente Mme Baujean Emma.

8. Ecrire une requête SQL permettant d'afficher le nom et le prénom de tous les patients du médecin M. ARNOS Pierre.

Partie C : la sécurité des mots de passe d'un médecin

Pour utiliser les ordinateurs de cet hôpital, tout le personnel doit saisir son identifiant puis son mot de passe. Pour davantage de sécurité, ce dernier doit être fort et changé régulièrement.

On appelle "mot de passe fort" une chaîne de caractères composée :

- d'au minimum 12 caractères ;
- d'au moins 2 majuscules ;
- d'au moins 2 chiffres ;
- d'au moins 2 symboles parmi **#@!?%<>=€$+-*/&**.

Par exemple, un médecin utilise le mot de passe fort : `@20!HôPiTaL&24#`.

On dispose des méthodes :

- `isalpha` qui permet de tester si un caractère est une lettre ;
- `isupper` qui permet de tester si un caractère est une majuscule ;
- `isdigit` qui permet de tester si un caractère est un chiffre.

```
>>> 'A'.isalpha()
True
>>> 'a'.isupper()
False
>>> '5'.isdigit()
True
```

De plus, on affecte tous les symboles dans une variable globale `liste_symboles` de type `str`:

```
liste_symboles = '#@!?%<>=€$+-*/&'
```

9. On considère une fonction `mdp_fort` qui prend en paramètre une chaîne de caractères `mdp` et qui renvoie `True` si `mdp` est un mot de passe fort et `False` sinon.
   Compléter le script de cette fonction `mdp_fort` en recopiant les lignes 2, 10, 12 et 14 sur votre copie :

```
1  def mdp_fort(mdp):
2      if ... :
3          return False
4      majuscules = 0
5      chiffres = 0
6      symboles = 0
7      for caractere in mdp :
8          if caractere.isupper():
9              majuscules+=1
10         if ... :
11             chiffres+=1
12         if ... :
```

```
13              symboles+=1
14      if ... :
15          return False
16      return True
```

Pour aider le personnel, le service informatique a mis en place une fonction `creation_mdp` permettant de générer aléatoirement un mot de passe.
Elle prend en paramètres quatre entiers de type `int` :

- la longueur `n` du mot de passe ;
- le nombre `nbr_m` de majuscules qu'il doit contenir au minimum ;
- le nombre `nbr_c` de chiffres qu'il doit contenir au minimum ;
- le nombre `nbr_s` de symboles qu'il doit contenir au minimum.

Cette fonction renvoie une chaîne de caractères respectant les conditions précédentes.

10. Compléter le script de cette fonction `creation_mdp` en recopiant les lignes 9 et de 13 à 19 sur votre copie.

```
1   def creation_mdp(n, nbr_m, nbr_c, nbr_s):
2       mdp = ''
3       caracteres='abcdefghijklmnopqrstuvwxyz' + \
4                  'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789' + \
5                  '#@!?%<>=€$+-*/&'
6       majuscules = 0
7       chiffres = 0
8       symboles = 0
9       while ... :
10          # la variable 'c' contient un caractère
11          # choisi aléatoirement dans la variable 'caracteres'
12          c = choice(caracteres)
13          if ... :
14              ...
15          if ... :
16              ...
17          if ... :
18              ...
19          mdp = ...
20      return mdp
```

Si un personnel n'utilise pas le générateur de mot de passe, il a tendance à y insérer des mots de la langue française.
Par exemple, le mot de passe `@20!HôPiTaL&24#` est fort mais on y trouve le mot "hôpital".
Si un mot de passe ne contient pas de mots de 4 lettres ou plus de la langue française ou étrangère, on dit que c'est un "mot de passe extra fort".

On dispose de la variable `dicoFR` (de type `list`) qui contient tous les mots de la langue française.

On dispose également de la fonction `transforme` qui prend en paramètre une chaîne de caractères et qui renvoie une nouvelle chaîne de caractère en transformant toutes les lettres majuscules en minuscules. Par exemple :

```
>>> transforme('@20!HôPiTaL&24#')
@20!hôpital&24#
```

Pour simplifier, on suppose que les mots présents dans un mot de passe sont entiers (pas d'abréviation) et sont séparés par des chiffres ou des symboles.

On considère une fonction `recherche_mot` qui prend en paramètre un mot de passe `mdp` (de type `str`) et qui renvoie des mots présents (de type `str`) dans `mdp`. Exemple :

```
>>> recherche_mot('@20!NeUrO&24#')
['neuro']
>>> recherche_mot('Chef!14NeuroA@85!')
['Chef','Neuro']
```

11. Compléter le script de cette fonction `recherche_mot` en recopiant les lignes 5, 6, 15, 16, 18 sur votre copie.

```
1  def recherche_mot(mdp):
2      mot = transforme(mdp)
3      trouve = []
4      i = 0
5      while ... :
6          if ... :  # si le caractère est un chiffre
7              i = i+1
8          elif mot[i] in liste_symboles:
9              i = i+1
10         else:
11            # si le caractère est une lettre, on prend les
12            # lettres qui la suivent jusqu'au moment où
13            # on trouve un chiffre ou un symbole
14             chaine = ''
15             while ... :
16                 chaine = ...
17                 i = i+1
18             ...
19     return trouve
```

12. Ecrire une fonction `mdp_extra_fort` qui prend en paramètre une chaîne de caractères `mdp` et qui renvoie `True` si `mdp` est un mot de passe extra fort et `False` sinon.