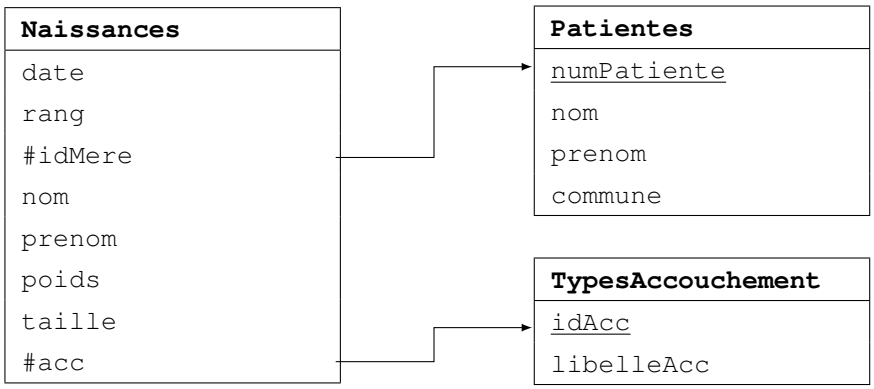
Amérique du sud - septembre 2022 - sujet 1

Exercice 1 (Bases de données - 4 points)
Le service maternité d'un hôpital utilise une base de données pour gérer les informations concernant les naissances qui y ont lieu.

Le schéma relationnel de cette base de données est le suivant :



Dans la relation `Naissances` :

- la clé étrangère `idMere` fait référence à la clé primaire `numPatiente` de la relation `Patientes` ;
- la clé étrangère `acc` fait référence à la clé primiaire `idAcc` de la relation `TypesAccouchement`. Il permet de savoir si la patiente a accouché par voie naturelle ou par césarienne ;
- l'attribut `rang` indique le rang de naissance du bébé dans le mois. Il recommence donc à 1 au début de chaque nouveau mois ;
- l'attribut `poids` est exprimé en grammes et l'attribut `taille` en centimètres.

Dans cet exercice, on pourra utiliser les mots-clés suivants du langage SQL : `SELECT`, `DELETE`, `FROM`, `WHERE`, `JOIN`, `INSERT`, `UPDATE`, `MIN`, `MAX`, `AVG`

Les fonctions d'agrégation `MIN(att)`, `MAX(att)` et `AVG(att)` renvoient respectivement la plus petite valeur, la plus grande valeur et la valeur moyenne de l'attribut `att` pour les enregistrements selectionnés. Ainsi, la requête

```
SELECT MIN(taille) FROM Naissances
```

renvoie la plus petite valeur de l'attribut `taille` de la table `Naissances`.

On donne ci-après des extraits des tables renvoyées par certaines requêtes SQL :

- `SELECT * FROM Naissances`

date	rang	idMere	nom	prenom	poids	taille	acc
...	...	...	...	...	...	...	...
28/02/2022	263	13857	Berthelot	Maïssa	3305	50	1
28/02/2022	264	13858	Samson	Pauline	3650	52	1
28/02/2022	265	13859	Perrin	Jonathan	3720	52	2
01/03/2022	1	13860	Fernandez	Lorette	3350	51	2
01/03/2022	2	13861	Baugé	Juliette	2870	48	1
01/03/2022	3	13861	Baugé	Noé	2985	49	1

- `SELECT * FROM Patientes`

numPatiente	nom	prenom	commune
...	...	...	...
13857	Berthelot	Michelle	Aigrefeuille d'Aunis
13858	Samson	Marine	Nieul sur Mer
13859	Perrin	Patricia	La Rochelle
13860	Fernandez	Claire	Aytré
13861	Baugé	Gaëlle	Lagord

⋆ `SELECT libelleAcc FROM TypesAccouchement`

libelleAcc
voie naturelle
césarienne

1. Indiquer, pour chacune des propositions suivantes, si elles peuvent être ou non clé primaire de la relation `Naissances`. Justifier chaque réponse.
   (a) `idMere`
   (b) `(date, rang)`
   (c) `(poids, taille)`
2. Pourquoi la requête ci-dessous provoque-t-elle une erreur ?

```
DELETE FROM Patientes WHERE numPatiente = 13858
```

3. Donner en langage SQL la requête d'insertion qui permet d'inscrire, avec le numéro 13862, la patiente Ninette Bélanger résidant à La Rochelle.
4. Dans l'extrait de la table fourni, on constate qu'une erreur a été commise lors de la saisie du prénom du bébé de madame Fernandez. La bonne orthographe est Laurette. Donner en langage SQL l'instruction à exécuter pour corriger le prénom du bébé.
5. Ecrire une requête SQL qui renvoie la liste des patientes (attributs `nom` et `prenom`) dont la commune de résidence est Aigrefeuille d'Aunis.
6. Ecrire une requête SQL qui renvoie le poids moyen des bébés nés par césarienne.
7. Quels sont les enregistrements renvoyés par la requête suivante lancée sur les extraits donnés des tables ?

```
SELECT Patientes.nom, Patientes.prenom
FROM Naissances
JOIN TypesAccouchement
ON Naissances.acc = TypesAccouchement.idAcc
JOIN Patientes
ON Naissances.idMere = Patientes.numPatiente
WHERE TypesAccouchement.idACC = 1
```

.

Exercice 2 (Programmation et tri - 4 points)
Au service des urgences d'un hôpital, le triage consiste à classifier ou à déterminer le degré de priorité des patients. Il implique une réévaluation périodique et systématique de ce degré pour les patients en attente.

Dans le système informatique, chaque patient obtient un identifiant à son arrivée en salle d'attente, ainsi qu'une priorité dépendant de la gravité potentielle de ses symptômes. Le patient ayant la priorité 1 est le premier qui doit être pris en charge et deux patients ne peuvent pas avoir la même priorité.

Pour modéliser la salle d'attente en Python, chaque patient est représenté par un tuple composé de son identifiant d'arrivée et de sa priorité. Ainsi, la variable `attente` implémentée ci-dessous représente une salle d'attente de trois patients où, à cet instant, le patient identifié 47 sera le premier à être pris en charge, puis le patient 45 et finalement le patient 49.

```
attente = [(45, 2), (47, 1), (49, 3)]
```

1. Ecrire l'instruction qui permet d'insérer dans la liste `attente` définie ci-dessus un nouveau patient identifié 50 avec une priorité de 4.
2. Pour optimiser le traitement informatisé de prise en charge des patients, on veut que la salle d'attente soit ordonnée par priorité. On utilise alors la fonction `tri` ci-dessous qui prend en paramètre une salle d'attente `attente` et qui renvoie cette salle triée dans l'ordre croissant des priorités.

```
def tri(attente):
    for i in range(len(attente)):
        pos = i
        mini = attente[i][1]
        for j in range(i, len(attente)):
            if attente[j][1] < mini:
                pos = j
                mini = attente[j][1]
        temp = attente[i]
        attente[i] = attente[pos]
        attente[pos] = temp
```

   (a) Quel agorithme de tri est ici implémenté ?
   i. le tri fusion ii. le tri par insertion iii. le tri par sélection iv. le tri rapide

   (b) Quelle est la complexité en temps des tris par insertion et par sélection ?
   i. constante : $O(1)$ ii. logarithmique : $O(\log n)$ iii. linéaire : $O(n)$ iv. quadratique : $O(n^2)$

3. On considère dans cette question que la salle d'attente est triée par ordre croissant des priorités.
   (a) Quand un patient est pris en charge, il faut l'enlever de la salle d'attente. La fonction `quitte`, qui prend en paramètre une salle d'attente `attente`, supprime le patient de priorité 1 de cette salle et renvoie la nouvelle salle d'attente.
   On aurait par exemple :

```
>>> quitte([(47, 1), (45, 2), (49, 3)])
[(45, 2), (49, 3)]
```

   Recopier et compléter le code ci-dessous à l'aide d'une liste par compréhension.

```
def quitte(attente):
    return [........................]
```

   (b) La fonction `quitte` ne met pas à jour les priorités des patients.
   Ecrire une fonction `maj` qui prend en paramètre une salle d'attente `attente` et qui met à jour, dans une nouvelle liste, les priorités des patients suite à la prise en charge d'un patient et renvoie cette nouvelle salle d'attente.
   On aurait par exemple :

```
>>> maj([(45, 2), (49, 3)])
[(45, 1), (49, 2)]
```

4. (a) Ecrire une fonction `priorite` prenant en paramètre une salle d'attente `attente` et un patient `p` et renvoyant l'ordre de priorité de ce patient.
On aurait par exemple :

```
>>> priorite([(45, 2), (47, 1), (49, 3)], 49)
3
```

(b) La fonction `revise` ci-dessous prend en paramètre une salle d'attente `attente` et un patient `p` et renvoie une nouvelle salle d'attente où le patient `p` dont la priorité médicale a évolué reçoit la priorité 1 et où les priorités des autres patients sont mises à jour.
On aurait par exemple :

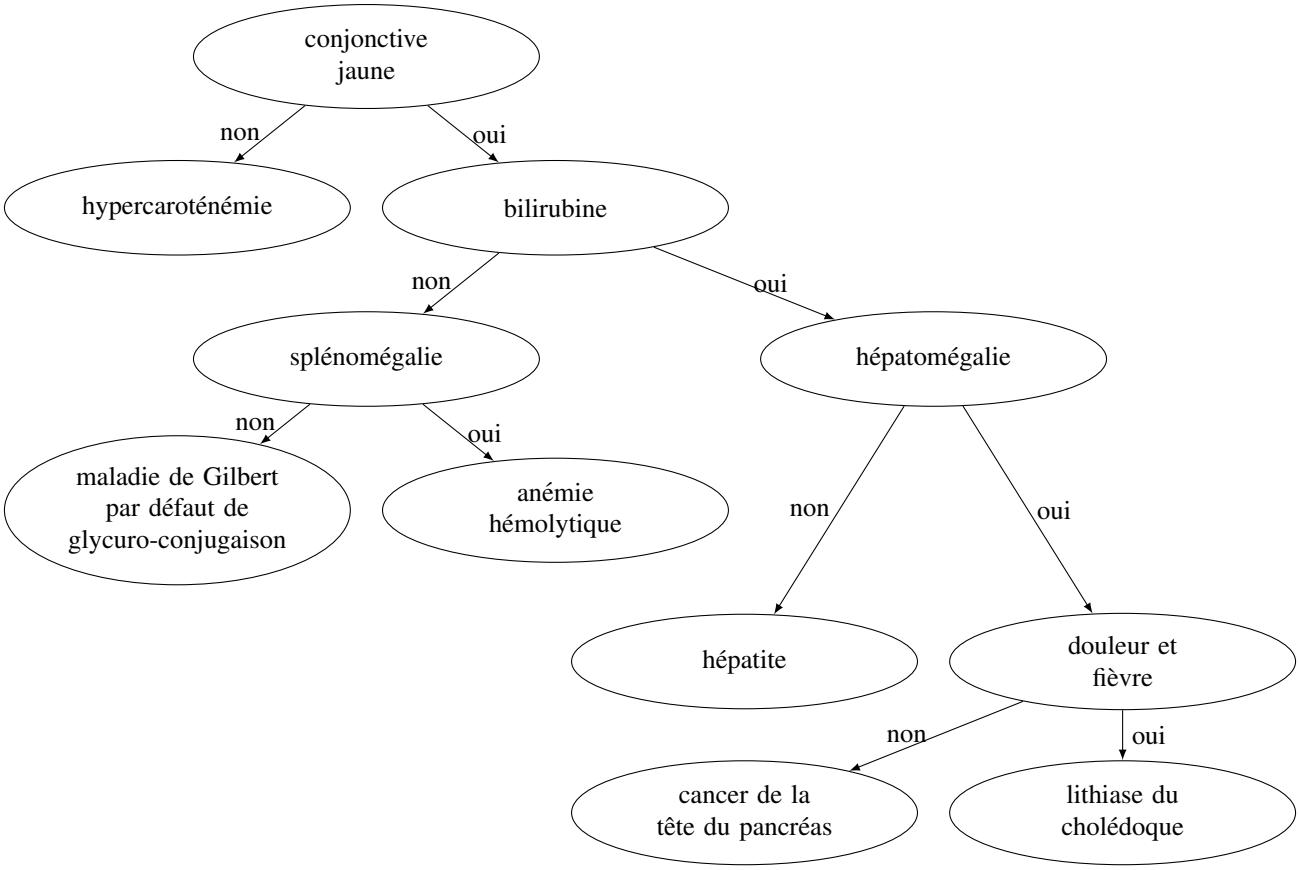
```
>>> revise([(45, 2), (47, 1), (49, 3)], 49)
[(45, 3), (47, 2), (49, 1)]
```

Recopier et compléter le code ci-dessous :

```
def revise(attente, p):
    nouvelle = []
    n = priorite(attente, p)
    for (patient, prio) in attente:
        if patient == p:
            nouvelle.append(.....................)
        elif ..................... :
            nouvelle.append((patient, prio + 1))
        else:
            nouvelle.append((patient, prio))
    return nouvelle
```

Exercice 3 (Arbres binaires - 4 points)

Les premiers travaux concernant l'aide à la décision médicale se sont développés pendant les années soixante-dix parallèlement à l'avènement de l'informatique dans le secteur médical. L'arbre de décision est une technique décisionnelle fréquemment employée pour rechercher la meilleure stratégie thérapeutique. L'arbre de décision de cet exercice, présenté ci-dessous, est un arbre binaire que l'on nommera `arb_decision`.



Arbre de décision et présence d'une jaunisse (peau anormalement jaune) chez un patient

Rappels :

- Un **arbre binaire** est une structure de données qui peut se représenter sous la forme d'une hiérarchie dont chaque élément, appelé **nœud**, porte une étiquette.
- Le nœud initial est appelé **racine**.
- Chaque nœud d'un arbre binaire possède au plus deux **sous-arbres**.
- Chacun de ces sous-arbres est un arbre binaire, appelés **sous-arbre gauche** et **sous-arbre droit**.
- Un nœud dont les sous-arbres sont vides est appelé une **feuille**.
- Dans cet exercice, on utilisera la définition suivante : la **hauteur** d'un arbre binaire ne comportant qu'un nœud est égale à 1.

Dans l'arbre de décision ci-dessus,

- un nœud représente un symptôme dont le médecin doit étudier la présence ou l'absence ; la réponse ne peut être que **oui** ou **non** ;
- le sous-arbre gauche d'un nœud donné décrit la démarche à adopter si le symptôme est **absent** ;
- le sous-arbre droit d'un nœud donné décrit la démarche à adopter si le symptôme est **présent** ;
- l'étiquette d'une feuille est la maladie induite par le chemin parcouru.

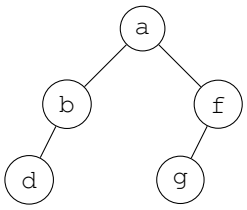
1. Déterminer la taille et la hauteur de l'arbre donné en exemple en introduction (arbre de décision en présence d'une jaunisse).
2. On choisit d'implémenter un arbre binaire à l'aide d'un dictionnaire

```
arbre_vide = {}
arbre = {'etiquette': valeur,
         'sag': sous_arbre_gauche,
         'sad': sous_arbre_droit}
```

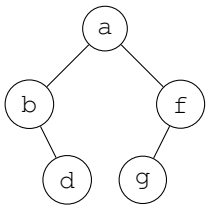
Le code ci-dessous représente un arbre selon le modèle précédent :

```
{'etiquette': 'a',
 'sag': {'etiquette': 'b',
         'sag': {},
         'sad': {'etiquette': 'd',
                 'sag': {},
                 'sad': {}}},
 'sad': {'etiquette': 'f',
         'sag': {'etiquette': 'g',
                 'sag': {},
                 'sad': {}},
         'sad': {}}}
```

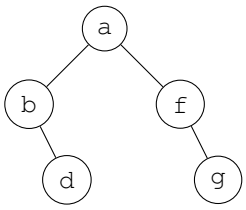
(a) A quelle représentation graphique correspond la structure implémentée ci-dessus ?



arbre 1



arbre 2



arbre 3

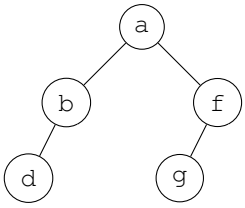
(b) Représenter graphiquement l'arbre correspondant au code ci-dessous :

```
{'etiquette': 'H',
 'sag': {'etiquette': 'G',
         'sag': {'etiquette': 'E',
                 'sag': {},
                 'sad': {}},
         'sad': {'etiquette': 'D',
                 'sag': {},
                 'sad': {'etiquette': 'B',
                         'sag': {},
                         'sad': {}}}},
 'sad': {'etiquette': 'F',
         'sag': {'etiquette': 'C',
                 'sag': {},
                 'sad': {'etiquette': 'A',
                         'sag': {},
                         'sad': {}}},
         'sad': {}}}
```

3. La fonction `parcours` ci-dessous prend en paramètre un arbre binaire `arb` et affiche les valeurs des étiquettes des différents nœuds de `arb`.

```
def parcours(arb):
    if arb == {}:
        return None
    parcours(arb['sag'])
    parcours(arb['sad'])
    print(arb['etiquette'])
```

(a) Donner l'affichage après l'appel de la fonction `parcours` avec l'arbre donné ci-dessous.



(b) Ecrire une fonction `parcours_maladie` prenant en paramètre un arbre de décision `arb` et n'affichant que les feuilles de `arb`, ce qui correspond aux maladies possiblement induites par `arb`.

4. On souhaite maintenant afficher l'ensemble des symptômes relatifs à une maladie. On considère pour cela une fonction `symptomes` prenant en paramètre un abre de décision `arb` et le nom `mal` d'une maladie. L'appel de cette fonction sur l'arbre de décision `arb_decision` de l'introduction fournit les affichages suivants :

```
>>> symptomes(arb_decision, 'anémie hémolytique')
sympômes de anémie hémolytique :
splénomégalie
pas de bilirubine
conjonctive jaune
```

Pour cela, on modifie la structure précédente en ajoutant une clé `surChemin` qui sera un booléen indiquant si le nœud est sur le chemin de la maladie. La clé `surChemin` est initialisée à `False` pour tous les nœuds :

```
arbre = {'etiquette': valeur,
         'surChemin': False,
         'sag': sous-arbre gauche,
         'sad': sous-arbre droit}
```

Recopier et compléter les lignes 6, 8, 14 et 18 du code suivant sur votre copie.

```
1  def symptomes(arb, mal):
2      if arb['sag'] != {}:
3          symptomes(arb['sag'], mal)
4
5      if arb['sad'] != {}:
6          symptomes(.....................)
7
8      if ..................... :
9          arb['surChemin'] = True
10         print('symptômes de', arb['etiquette'], ':')
11
12     else:
13         if arb['sad'] != {} and arb['sad']['surChemin']:
14             print(.....................)
15             arb['surChemin'] = True
16
17         if arb['sag'] != {} and arb['sag']['surChemin']:
18             print(.....................)
19             arb['surChemin'] = True
```

Exercice 4 (Processus et OS - 4 points)
Les parties A et B peuvent être traitées indépendamment.

A. Ordonnancement des processus.
Dans le laboratoire d'analyse médicale d'un hôpital, plusieurs processus peuvent demander l'allocation du processeur simultanément.

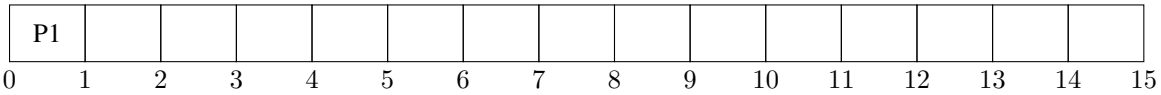
Le tableau ci-dessous donne les demandes d'exécution de quatre processus et indique :

- le temps d'exécution du processus (unité de temps) ;
- l'instant d'arrivée du processus sur le processeur (unité de temps) ;
- le numéro de priorité du processus (classé de 1 à 10)

Plus la priorité est grande, plus le numéro de priorité est petit. Ainsi, le processus P3, du tableau ci-dessous, est plus prioritaire que le processus P1.
L'ordonnancement est de type préemptif, ce qui signifie qu'à chaque unité de temps, le processeur choisit d'exécuter le processus ayant le plus petit numéro de priorité (un seul processus à la fois). Ceci peut provoquer la suspension d'un autre processus qui reprendra lorsqu'il deviendra le plus prioritaire dans la file d'attente.

Processus	Temps d'exécution	Instant d'arrivée	Numéro de priorité
P1	3	0	4
P2	4	2	2
P3	3	3	1
P4	4	5	3

1. Reproduire le diagramme ci-dessous sur votre copie et indiquer dans chacune des cases le processus exécuté par le processeur entre deux unités de temps (il peut y avoir des cases vides).



2. Recopier et compléter les temps de séjour ainsi que les temps d'attente de chacun des processus (toujours en unités de temps). On rappelle que
   - Temps de séjour = instant de terminaison − instant d'arrivée
   - Temps d'attente = temps de séjour - temps d'exécution

Processus	Temps d'exécution	Instant d'arrivée	Temps de séjour	Temps d'attente
P1	3	0	$14-0=14$	$14-3=11$
P2	4	2		
P3	3	3		
P4	4	5		

3. A quelles conditions le temps d'attente d'un processus peut-il être nul ?

B. Processus et ressources.
Dans ce laboratoire d'analyse médicale de l'hôpital, le laborantin en charge du traitement des différents prélèvements (sanguins, urinaires et biopsiques) utilise simultanément quatre logiciels :

- un logiciel d'analyse d'échantillons (connecté à l'analyseur) ;
- un logiciel d'accès à la base de données des patients (SGBD) ;
- un traitement de texte ;
- un tableur.

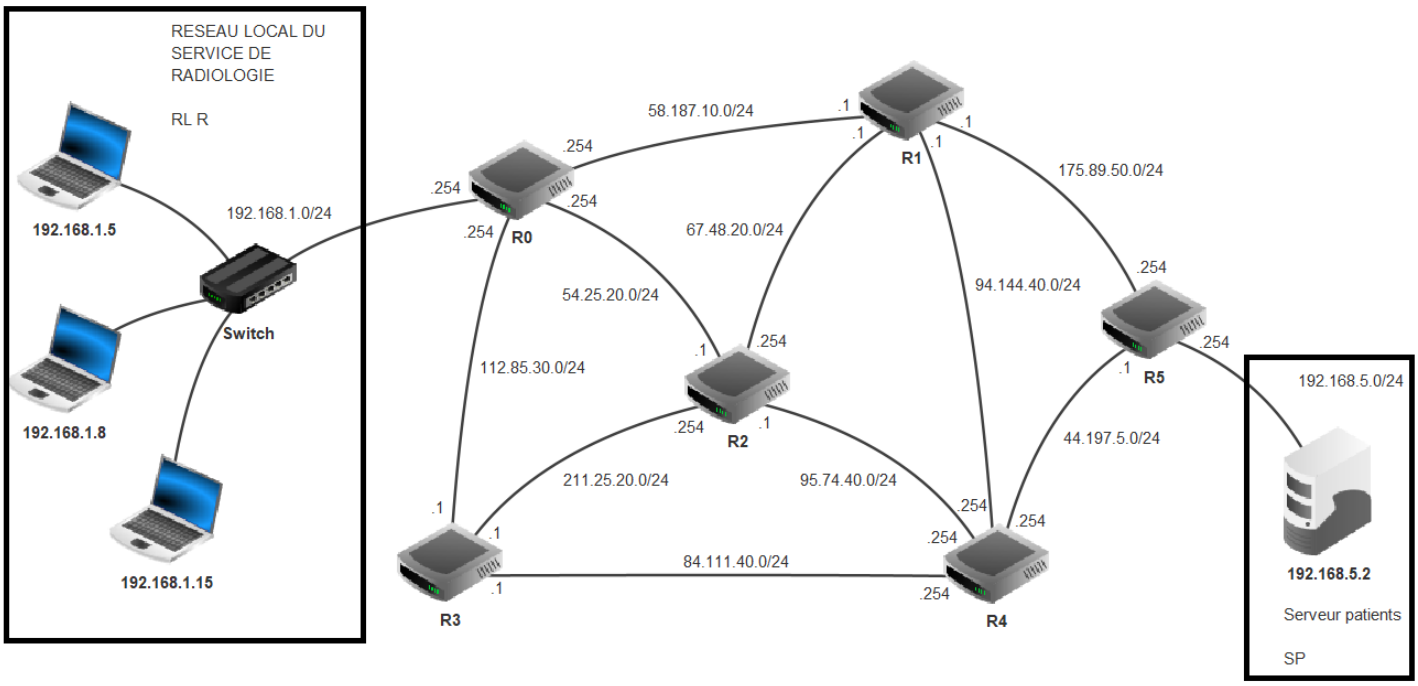
Le tableau ci-dessous donne l'état à un instant donné des différents processus (instances des programmes) qui peuvent soit mobiliser (M) des données (D1, D2, D3, D4 ou D5), soit être en attente des données (A) ou ne pas les solliciter (-).
Une donnée ne peut être mobilisée que par un seul processus à la fois. Si un autre processus demande une donnée déjà mobilisée, il passe en attente.
Par exemple, le SGBD mobilise la donnée D4 et est en attente de la donnée D5.

	D1	D2	D3	D4	D5
Analyseur d'échantillons	M	-	-	A	-
SGBD	-	-	-	M	A
Traitement de texte	-	M	A	-	-
Tableur	A	-	M	-	M

1. A partir du tableau ci-dessus, démontrer que, à cet instant, les processus s'attendent mutuellement.
2. Comment s'appelle cette situation ?
3. On suppose que l'analyseur d'échantillon libère la ressource D1. Donner un ordre possible d'exécution des processus.

Exercice 5 (Réseaux et protocoles de routage - 4 points)
Cet exercice comporte trois parties A, B et C.

Un extrait de l'architecture réseau d'un centre hospitalier est présenté sur le schéma ci-dessous. Celui-ci met en évidence le réseau local du service de radiologie, nommé RL R, et le serveur de données des patients inscrits à la sécurité sociale, nommé SP.
Dans ce réseau, R0, R1, R2, R3, R4 et R5 représentent des routeurs.



Les adresses et les masques réseau sont indiqués à côté de chacune des connexions sous la forme `X1.X2.X3.X4/n`, où `X1`, `X2`, `X3` et `X4` représentent les quatre octets de l'adresse IP et où `n` représente le nombre de bits égaux à 1 dans le asque. On rappelle qu'un masque est consitué de 32 bits dont les n premiers bits sont à 1 et les autres à 0. Celui-ci définit avec l'adresse réseau une plage d'adresses IP dont :

- les n premiers bits, appelés « partie réseau » sont fixes ;
- les bits restants, formant la « partie machine » peuvent prendre toutes les valeurs possibles.

Les adresses IP de toutes les machines connectées à un même réseau ont donc la même partie réseau. Enfin, deux adresses IP ne peuvent être attribuées à une machine :

- celle dont tous les bits de la partie machine sont à 0 (adresse réseau) ;
- celle dont tous les bits de la partie machine sont à 1 (adresse de diffusion).

Le repérage au niveau des interfaces de connexion des différents routeurs permet de connaître l'adresse utilisée par le routeur (passerelle) en fonction de l'adresse réseau. Exemples pour la liaison entre le routeur R0 et le routeur R1 :

- Adresse de l'interface de R0 qui permet de communiquer avec R1 : `58.187.10.254`
- Adresse de l'interface de R1 qui permet de communiquer avec R0 : `58.187.10.1`

A. Adressage.

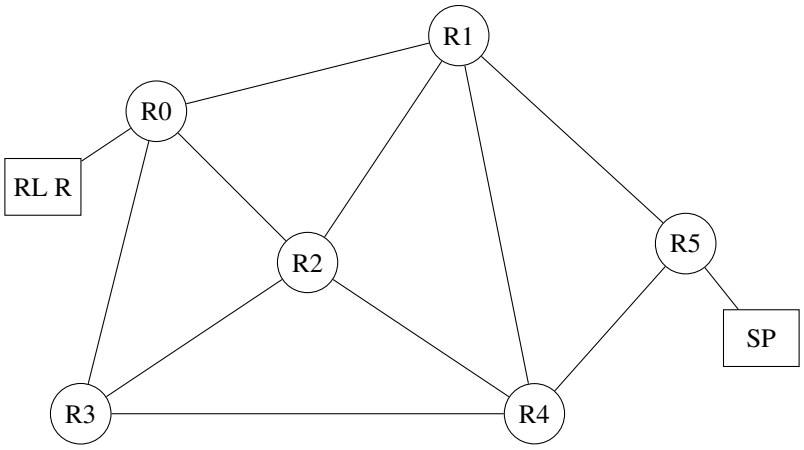
1. Quels sont l'adresse et le masque du réseau local du service de radiologie (RL R) ?
2. Donner les adresses des trois interfaces du routeur R5 permettant de transmettre ou de recevoir des données.
3. (a) Donner la première et la dernière adresse IP pouvant être attribuée à une machine sur le réseau RL R.
   (b) En déduire le nombre de machines pouvant être connectées sur ce réseau.

L'administrateur réseau du centre hospitalier souhaite statuer sur les performances de deux protocoles de routage. Il étudie donc la transmission de l'information depuis le serveur patients (SP) vers le service de radiologie (RL R).
L'étude portera donc sur les chemins entre les routeurs R5 et R0.

B. Etude du protocole RIP (Routing Information Protocol).
Dans cette partie, tous les routeurs utilisent le protocole RIP (distance en nombre de sauts).

1. Le serveur SP doit transmettre des données au service de radiologie (via le routeur R0) en effectuant le moins de sauts possibles. Citer les routeurs parcourus par le paquet.
2. Suite à une opération de maintenance, le serveur R1 est déconnecté. Plus aucun paquet ne peut transiter par ce routeur. Déterminer une nouvelle route empruntée par les paquets en citant les routeurs dans l'ordre.

C. Protocole OSPF (Open Shortest Path First).
Le serveur R1 est reconnecté au réseau et est fonctionnel.
Maintenant, pour tenir compte du débit des liaisons, l'administrateur réseau décide d'étudier le protocole OSPF (distance liée au coût des liaisons) pour effectuer le routage.



Les bandes passantes (BP) ainsi que le coût des différentes liaisons sont données dans les tableaux suivants :

Liaison entre R0 et :		
	BP	Coût
R1	500 Mb/s	2
R2	100 Mb/s	10
R3	300 Mb/s	4

Liaison entre R1 et :		
	BP	Coût
R2	10 Gb/s	1
R4	100 Mb/s	10
R5	100 Mb/s	10

Liaison entre R2 et :		
	BP	Coût
R3	400 Mb/s	
R4	300 Mb/s	4

Liaison entre R4 et :		
	BP	Coût
R3		5
R5	1 Gb/s	1

Pour calculer le coût d'une liaison, on utilise la formule :

$$C = \frac{\text{bande passante de référence}}{\text{bande passante de la liaison}} = \frac{10^9}{\text{BP}}$$

où BP est la bande passante de la connexion en b/s (bit par seconde).

Si le résultat du calcul n'est pas un entier, le coût est la valeur entière immédiatement supérieure. Par exemple, pour le calcul du coût entre R1 et R2 :

$$C = \frac{10^9}{10 \times 10^9} = 0{,}1$$

donc le coût est de 1.

1. Calculer le coût de la liaison entre R2 et R3.
2. Donner une bande passante possible de la connexion entre R3 et R4.
3. Déterminer le chemin parcouru par un paquet partant du serveur patients (SP) vers le service de radiologie (RL R) en utilisant le protocole OSPF. On précisera également le coût de ce chemin.
4. Suite à une opération de maintenance, la liaison R0-R1 est déconnectée : plus aucun paquet ne peut transiter par cette liaison. Déterminer une nouvelle route empruntée par les paquets en citant les routeurs dans l'ordre.