

# Métropole - mars 2021 - sujet 2

## Exercice 1 (Arbre binaire de recherche et POO - 4 points)

Une agence immobilière développe un programme pour gérer les biens immobiliers qu'elle propose à la vente.

Dans ce programme, pour modéliser les données de biens immobiliers, on définit une classe `Bim` avec les attributs suivants :

- \* `nt` de type `str` représente la nature du bien (appartement, maison, bureau, commerces, ...);
- \* `sf` de type `float` est la surface du bien;
- \* `pm` de type `float` est le prix moyen par  $m^2$  du bien qui dépend de son emplacement.

La classe `Bim` possède une méthode `estim_prix` qui renvoie une estimation du prix du bien. Le code (incomplet) de la classe `Bim` est donné ci-dessous :

```
class Bim:
    def __init__(self, nature, surface, prix_moy):
        ...

    def estim_prix(self):
        return self.sf * self.pm
```

1. Recopier et compléter le code du constructeur de la classe `Bim`.
2. On exécute l'instruction suivante : `b1 = Bim('maison', 70.0, 2000.0)`  
Que renvoie l'instruction `b1.estim_prix()` ? Préciser le type de la valeur renvoyée.
3. On souhaite affiner l'estimation du prix d'un bien en prenant en compte sa nature :
  - \* pour un bien dont l'attribut `nt` est 'maison' la nouvelle estimation du prix est le produit de sa surface par le prix moyen par  $m^2$  multiplié par 1,1;
  - \* pour un bien dont l'attribut `nt` est 'bureau' la nouvelle estimation du prix est le produit de sa surface par le prix moyen par  $m^2$  multiplié par 0,8;
  - \* pour les biens d'autres natures, l'estimation du prix ne change pas.

Modifier le code de la méthode `estim_prix` afin de prendre en compte ce changement de calcul.

4. Ecrire le code Python d'une fonction `nb_maison(lst)` qui prend en argument une liste Python de biens immobiliers de type `Bim` et qui renvoie le nombre d'objets de nature 'maison' contenus dans la liste `lst`.
5. Pour une recherche efficace des biens immobiliers selon le critère de leur surface, on stocke les objets de type `Bim` dans un arbre binaire de recherche, nommé `abr`. Pour tout nœud de cet arbre :
  - \* tous les objets de son sous-arbre gauche ont une surface inférieure ou égale à la surface de l'objet contenue dans ce nœud;
  - \* tous les objets de son sous-arbre droit ont une surface strictement supérieure à la surface de l'objet contenue dans ce nœud.

L'objet `abr` dispose des méthodes suivantes :

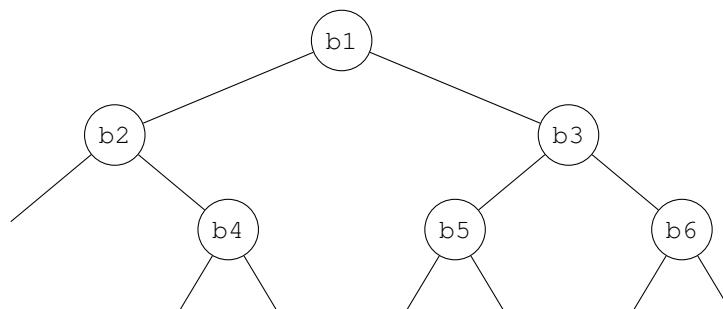
`abr.est_vide()` : renvoie `True` si `abr` est vide et `False` sinon.

`abr.get_v()` : renvoie l'élément (de type `Bim`) situé à la racine de `abr` si `abr` n'est pas vide et `None` sinon.

`abr.get_g()` : renvoie le sous-arbre gauche de `abr` si `abr` n'est pas vide et `None` sinon.

`abr.get_d()` : renvoie le sous-arbre droit de `abr` si `abr` n'est pas vide et `None` sinon.

- (a) Dans cette question, on suppose que l'arbre binaire `abr` a la forme ci-dessous :



Donner la liste des biens b1, b2, b3, b4, b5, b6 triée dans l'ordre croissant de leur surface.

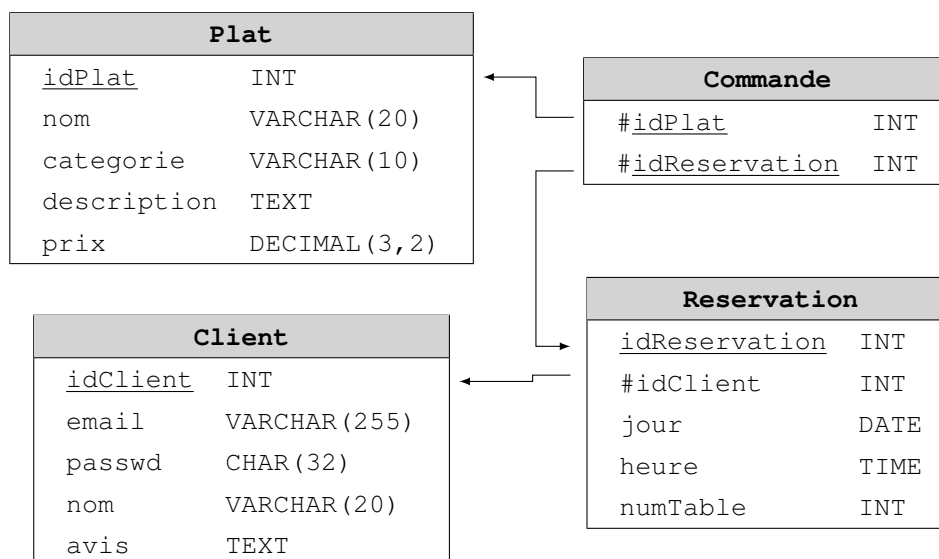
- (b) Recopier et compléter le code de la fonction récursive `contient` donnée ci-dessous, qui prend en arguments un nombre `surface` de type `float` et un arbre binaire de recherche `abr` contenant des éléments de type `Bim` ordonnés selon leur attribut de surface `sf`. La fonction `contient(surface, abr)` renvoie `True` s'il existe un bien dans `abr` d'une surface supérieure ou égale à `surface` et `False` sinon.

```
def contient(surface, abr):  
    if abr.est_vide():  
        return False  
    elif abr.get_v().sf >= .....:  
        return True  
    else:  
        return contient(surface, .....)
```

**Exercice 2 (SQL - 4 points)**

Une restauratrice a mis en place un site Web pour gérer ses réservations en ligne. Chaque client peut s'inscrire en saisissant ses identifiants. Une fois connecté, il peut effectuer une réservation en renseignant le jour et l'heure. Il peut également commander son menu en ligne et écrire un avis sur le restaurant.

Le gestionnaire du site Web a créé une base de données associée au site nommée *restaurant*, contenant les quatre relations du schéma relationnel ci-dessous :



Dans le schéma relationnel précédent, un attribut souligné indique qu'il s'agit d'une clé primaire. Un attribut précédé du symbole # indique qu'il s'agit d'une clé étrangère et la flèche associée indique l'attribut référencé. Ainsi, par exemple, l'attribut *idPlat* de la relation *Commande* est une clé étrangère qui fait référence à l'attribut *idPlat* de la relation *Plat*.

Dans la suite, les mots clés suivants du langage SQL pourront être utilisés dans les requêtes :

SELECT, FROM, WHERE, JOIN, ON, DELETE, UPDATE, SET, INSERT INTO, AND, OR.

1. Parmi les trois requêtes suivantes, écrites dans le langage SQL, laquelle renvoie les valeurs de tous les attributs des plats de la catégorie 'entrée' :

R1: **SELECT** nom, prix  
**FROM** Plat  
**WHERE** categorie = 'entrée'

R2: **SELECT** \*  
**FROM** Plat  
**WHERE** categorie = 'entrée'

R3: **UPDATE** Plat  
**SET** categorie = 'entrée'  
**WHERE** 1

2. Ecrire, dans le langage SQL, des requêtes d'interrogation sur la base de données restaurant permettant de réaliser les tâches suivantes :

- (a) Afficher les noms et les avis des clients ayant effectué une réservation pour la date du '2021-06-05' à l'heure '19:30:00'.
- (b) Afficher le nom des plats des catégories 'plat principal' et 'dessert' correspondant aux commandes de la date '2021-04-12'.

3. Que réalise la requête SQL suivante ?

```

INSERT INTO Plat
VALUES (58, 'Pêche Melba', 'dessert', 'Pêches et glace vanille', 6.5)
  
```

4. Ecrire des requêtes SQL permettant de réaliser les tâches suivantes :

- (a) Supprimer les commandes ayant comme *idReservation* la valeur 2047.
- (b) Augmenter de 5% tous les prix de la relation *plat* strictement inférieurs à 20.00.

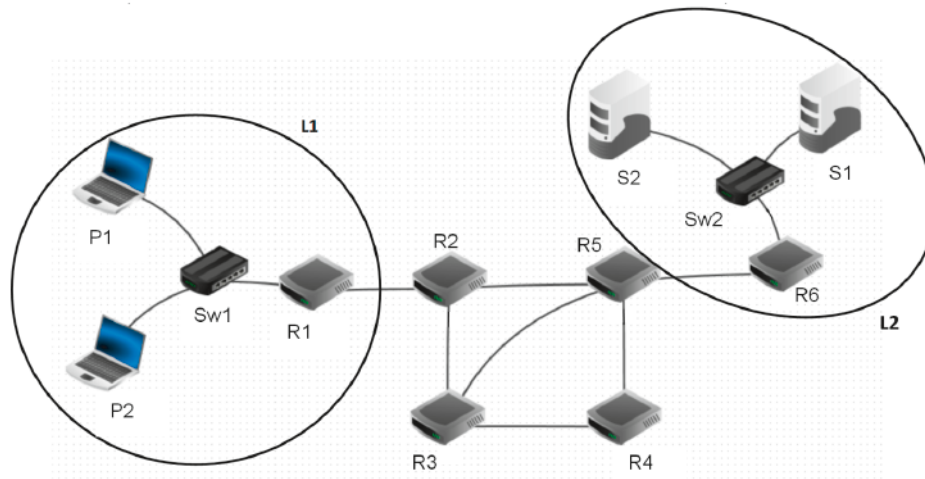
**Exercice 3 (Réseaux et protocoles de routage - 4 points)**

FIGURE 1 – Réseau d’une entreprise

La figure 1 ci-dessus représente le schéma d’un réseau d’une entreprise. Il y figure deux réseaux locaux L1 et L2. Ces deux réseaux locaux sont interconnectés par les routeurs R2, R3, R4 et R5. Le réseau local L1 est constitué des PC portables P1 et P2 connectés à la passerelle R1 par le switch Sw1. Les serveurs S1 et S2 sont connectés à la passerelle R6 par le switch Sw2.

La table 1 suivante indique les adresses IPv4 des machines constituant le réseau de l’entreprise.

Nom	Type	Adresse IPv4
R1	routeur (passerelle)	Interface 1 : 192.168.1.1/24 Interface 2 : 10.1.1.2/24
R2	routeur	Interface 1 : 10.1.1.1/24 Interface 2 : 10.1.2.1/24 Interface 3 : 10.1.3.1/24
R3	routeur	Interface 1 : 10.1.2.2/24 Interface 2 : 10.1.4.2/24 Interface 3 : 10.1.5.2/24
R4	routeur	Interface 1 : 10.1.5.1/24 Interface 2 : 10.1.6.1/24
R5	routeur	Interface 1 : 10.1.3.2/24 Interface 2 : 10.1.4.1/24 Interface 3 : 10.1.6.2/24 Interface 4 : 10.1.7.1/24
R6	routeur (passerelle)	Interface 1 : 172.16.0.1/16 Interface 2 : 10.1.7.2/24
P1	ordinateur portable	192.168.1.40/24
P2	ordinateur portable	192.168.1.46/24
S1	serveur	172.16.8.10/16
S2	serveur	172.16.9.12/16

TABLE 1 – Adresses IPv4 des machines

**Rappels et notations.**

Rappelons qu’une adresse IP est composée de 4 octets, soit 32 bits. Elle est notée  $X1.X2.X3.X4$ , où  $X1$ ,  $X2$ ,  $X3$  et  $X4$  sont les valeurs des 4 octets. Dans la table 1, les valeurs des 4 octets ont été converties en notation décimale.

La notation  $X1.X2.X3.X4/n$  signifie que les  $n$  premiers bits de poids forts de l’adresse IP représentent la partie « réseau », les bits suivants de poids faibles représentent la partie « machine ».

Toutes les adresses des machines connectées à un réseau local ont la même partie réseau. L’adresse IP dont tous les bits de la partie « machine » sont à 0 est appelée « adresse du réseau ». L’adresse IP dont tous les bits de la partie « machine » sont à 1 est appelée « adresse de diffusion ».

- (a) Quelles sont les adresses des réseaux locaux L1 et L2 ?

- (b) Donner la plus petite et la plus grande adresse IP valides pouvant être attribuées à un ordinateur portable ou un serveur sur chacun des réseaux L1 et L2 sachant que l'adresse du réseau et l'adresse de diffusion ne peuvent pas être attribuées à une machine.
- (c) Combien de machines peut-on connecter au maximum à chacun des réseaux locaux L1 et L2 ? On donne ci-dessous les valeurs de quelques puissances de 2 ?

$2^6$	$2^7$	$2^8$	$2^9$	$2^{10}$	$2^{11}$	$2^{12}$	$2^{13}$	$2^{14}$	$2^{15}$	$2^{16}$	$2^{17}$
64	128	256	512	1024	2048	4096	8192	16384	32768	65536	131072

2. (a) Expliquer l'utilité d'avoir plusieurs chemins possibles reliant les réseaux L1 et L2.
- (b) Quel est le chemin le plus court en nombre de sauts pour relier R1 et R6 ? Donner le nombre de sauts de ce chemin et préciser les routeurs utilisés.
- (c) La bande passante d'une liaison Ether (quantité d'information qui peut être transmise en bits/s) est de 107 bits/s et celle d'une liaison FastEther est de 108 bits/s. Le coût d'une liaison est défini par  $10^8/d$ , où  $d$  est sa bande passante en bits/s.

Liaison	R1-R2	R2-R5	R5-R6	R2-R3	R3-R4	R4-R5	R3-R5
Type	Ether	Ether	Ether	FastEther	FastEther	FastEther	Ether

TABLE 2 – Type des liaisons entre les routeurs

Quel est le chemin reliant R1 et R6 qui a le plus petit coût ? Donner le coût de ce chemin et préciser les routeurs utilisés.

3. Dans l'annexe A figurent les tables de routages des routeurs R1, R2, R5 et R6 au démarrage du réseau. Indiquer sur votre copie ce qui doit figurer dans les lignes laissées vides des tables de routage des routeurs R5 et R6 pour que les échanges entre les ordinateurs des réseaux L1 et L2 se fassent en empruntant le chemin le plus court en nombre de sauts.

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

Dans un bureau d'architectes, on dispose de certaines ressources qui ne peuvent être utilisées simultanément par plus d'un processus, comme l'imprimante, la table traçante, le modem. Chaque programme, lorsqu'il s'exécute, demande l'allocation des ressources qui lui sont nécessaires. Lorsqu'il a fini de s'exécuter, il libère ses ressources.

Programme 1	Programme 2	Programme 3
demander(table traçante)	demander(modem)	demander(imprimante)
demander(modem)	demander(imprimante)	demander(table traçante)
exécution	exécution	exécution
libérer(modem)	libérer(imprimante)	libérer(table traçante)
libérer(table traçante)	libérer(modem)	libérer(imprimante)

On appelle p1, p2 et p3 les processus associés respectivement aux programmes 1, 2 et 3.

1. Les processus s'exécutent de manière concurrente. Justifier qu'une situation d'interblocage peut se produire.
2. Modifier l'ordre des instructions du programme 3 pour qu'une telle situation ne puisse pas se produire. Aucune justification n'est attendue.
3. Supposons que le processus p1 demande la table traçante alors qu'elle est en cours d'utilisation par le processus p3. Parmi les états suivants, quel sera l'état du processus p1 tant que la table traçante n'est pas disponible :
 

(a) élu
(b) bloqué
(c) prêt
(d) terminé

**Partie B.**

Avec une ligne de commande dans un terminal sous Linux, on obtient l'affichage suivant :

UID	PID	PPID	C	STIME	TTY	TIME	CMD
...							
pi	6211	831	8	09:07	?	00:01:16	/usr/lib/chromium-browser/chromium-browser-v7 --disable-quic -enable-tcp-fast-open --
pi	6252	6211	0	09:07	?	00:00:00	/usr/lib/chromium-browser/chromium-browser-v7 --type=zygote --ppapi-flash-path=/usr
pi	6254	6252	0	09:07	?	00:00:00	/usr/lib/chromium-browser/chromium-browser-v7 --type=zygote --ppapi-flash-path=/usr
pi	6294	6211	4	09:07	?	00:00:40	/usr/lib/chromium-browser/chromium-browser-v7 --type=gpu-process --filed-trial-handle=
pi	6300	6211	1	09:07	?	00:00:16	/usr/lib/chromium-browser/chromium-browser-v7 --type=utility --field-trial-handle=10758
pi	6467	6254	1	09:07	?	00:00:11	/usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi	11267	6254	2	09:12	?	00:00:15	/usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi	12035	836	0	09:13	?	00:00:00	/usr/lib/libreoffice/program/oosplash --writer file:///home/pi/Desktop/mon.fichier.odt
pi	12073	12035	2	09:13	?	00:00:15	/usr/lib/libreoffice/program/soffice.bin --writer file:///home/pi/Desktop/mon.fichier.odt
pi	12253	831	1	09:13	?	00:00:07	/usr/bin/python3 /usr/bin/sense.emu.gui
pi	20010	6211	1	09:21	?	00:00:00	/usr/lib/chromium-browser/chromium-browser-v7 --type=utility --field-trial-handle=10758
pi	20029	6254	56	09:21	?	00:00:28	/usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi	20339	6254	4	09:21	?	00:00:01	/usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi	20343	6254	2	09:21	?	00:00:00	/usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi	20464	6211	17	09:22	?	00:00:00	/proc/self/exe --type=utility --field-trial-handle=1075863133478894917,6303120996223181
pi	20488	6254	14	09:22	?	00:00:00	/usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi	20519	676	0	09:22	pts/0	00:00:00	ps -ef

La documentation Linux donne la signification des différents champs :

- \* UID : identifiant utilisateur effectif ;
- \* PID : identifiant de processus ;
- \* PPID : PID du processus parent ;
- \* C : partie entière du pourcentage d'utilisation du processeur par rapport au temps de vie des processus ;
- \* STIME : l'heure de lancement du processus ;
- \* TTY : terminal de contrôle
- \* TIME : temps d'exécution
- \* CMD : nom de la commande du processus

1. Parmi les quatre commandes suivantes, laquelle a permis cet affichage ?
 

(a) `ls -l`
(c) `cd ..`

(b) `ps -ef`
(d) `chmod 741 processus.txt`
2. Quel est l'identifiant du processus parent à l'origine de tous les processus concernant le navigateur Web (chromium-browser) ?
3. Quel est l'identifiant du processus dont le temps d'exécution est le plus long ?

**Exercice 5 (Piles et files - 4 points)**

Une méthode simple pour gérer l'ordonnancement des processus est d'exécuter les processus en une seule fois et dans leur ordre d'arrivée.

1. Parmi les propositions suivantes, quelle est la structure de données la plus appropriée pour mettre en oeuvre le mode FIFO (First In First Out) ?

(a) liste (b) dictionnaire (c) pile (d) file

2. On choisit de stocker les données des processus en attente à l'aide d'une liste Python `lst`.

On dispose déjà d'une fonction `retirer` qui prend en argument une liste `lst` et qui renvoie l'élément `lst[0]` puis le supprime de la liste `lst`.

Ecrire en Python le code d'une fonction `ajouter` qui prend en argument une liste `lst` et un processus `proc` et qui ajoute à la fin de la liste `lst` le nouveau processus en attente `proc`.

On choisit maintenant d'implémenter une file `file` à l'aide d'un couple  $(p1, p2)$  où `p1` et `p2` sont des piles. Ainsi `file[0]` et `file[1]` sont respectivement les piles `p1` et `p2`.

Pour enfiler un nouvel élément `elt` dans `file`, on l'empile dans `p1`.

Pour défiler `file`, deux cas se présentent :

- La pile `p2` n'est pas vide : on dépile `p2`.
- La pile `p2` est vide : on dépile les éléments de `p1` en les empilant dans `p2` jusqu'à ce que `p1` soit vide, puis on dépile `p2`.

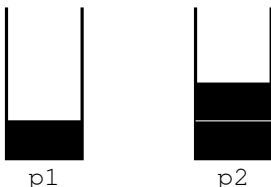

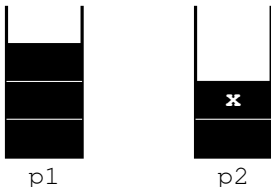
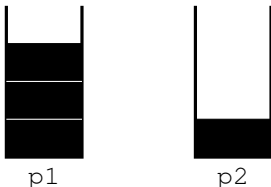
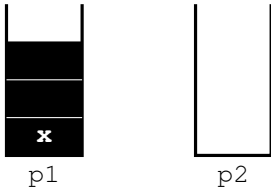
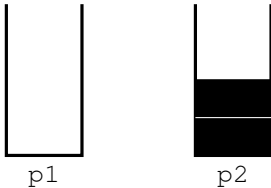
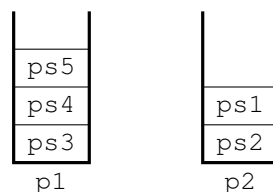
	Etat de la file <code>file</code> avant	Etat de la file <code>file</code> après
<code>enfiler(file, elt)</code>	 <p>p1 p2</p>	 <p>p1 p2</p>
<code>defiler(file)</code> cas où <code>p2</code> n'est pas vide	 <p>p1 p2</p>	 <p>p1 p2</p>
<code>defiler(file)</code> cas où <code>p2</code> est vide	 <p>p1 p2</p>	 <p>p1 p2</p>

Illustration du fonctionnement des fonctions `enfiler` et `defiler`

3. On considère la situation représentée ci-dessous :



On exécute la séquence d'instructions suivante :

```
enfiler(file, ps6)
defiler(file)
defiler(file)
defiler(file)
enfiler(file, ps7)
```

Représenter le contenu final des deux piles à la suite de ces instructions.

4. On dispose des fonctions suivantes :

- `empiler` qui prend en arguments une pile `p` et un élément `elt`, et qui empile l'élément `elt` dans la pile `p`,
  - `depiler` qui prend en argument une pile `p` et qui renvoie le sommet de la pile `p` si `p` n'est pas vide et le supprime,
  - `pile_vide` qui prend en argument une pile `p` et qui renvoie `True` si la pile `p` est vide, `False` si la pile `p` n'est pas vide.
- (a) Ecrire en Python une fonction `est_vide` qui prend en argument un couple de piles `f` et qui renvoie `True` si la file représentée par `f` est vide, `False` sinon.
- (b) Ecrire en Python une fonction `enfiler` qui prend en arguments un couple de piles `f` et un élément `elt` et qui ajoute `elt` en queue de la file représentée par `f`.
- (c) Ecrire en Python une fonction `defiler` qui prend en argument un couple de piles `f` et qui renvoie, si la file représentée par `f` est non vide, l'élément en tête `f` en le retirant. Une erreur doit être signalée si `f` est vide.

## Annexe A de l'exercice 3

## Tables de routage du réseau de la figure 2

IP réseau de destination	Passerelle suivante	Interface
192.168.1.0/24	192.168.1.1/24	Interface 2
10.1.1.0/24	10.1.1.2/24	Interface 1
0.0.0.0/0	10.1.1.1/24	Interface 1

Table de routage du routeur R1

IP réseau de destination	Passerelle suivante	Interface
10.1.1.0/24	10.1.1.1/24	Interface 1
10.1.2.0/24	10.1.2.1/24	Interface 2
10.1.3.0/24	10.1.3.1/24	Interface 3
192.168.1.0/24	10.1.1.2/24	Interface 2
172.16.0.0/16	10.1.3.2/24	Interface 3

Table de routage du routeur R2

IP réseau de destination	Passerelle suivante	Interface
10.1.3.0/24	10.1.3.2/24	Interface 1
10.1.4.0/24	10.1.4.2/24	Interface 2
10.1.6.0/24	10.1.6.2/24	Interface 3
10.1.7.0/24	10.1.7.1/24	Interface 4

Table de routage du routeur R5

IP réseau de destination	Passerelle suivante	Interface
172.16.0.0/16	172.16.0.1/16	Interface 1

Table de routage du routeur R6