

DS13

Tables de données

Réseaux de télécommunications

Durée de l'épreuve : 01h30

L'usage de la calculatrice n'est pas autorisé.

Le candidat répond sur feuilles doubles numérotées et garde l'énoncé.

Les traces de recherche, même incomplètes ou infructueuses, seront valorisées.

La qualité de la rédaction, la clarté et la précision des raisonnements seront prises en compte.

QUESTION 1

Programmer une fonction ***importTable*** qui prend en paramètre le nom d'un fichier texte en format **csv** et renvoie la table de données correspondante sous la forme d'une liste de dictionnaires :

```
# fichier "élèves.csv"
INE;Nom;Prénom;
1;Nagoumi;Lim;
2;Chagoumi;Zaza;
3;Ghazoumi;Didi;
4;Berroumi;Lili;
```

```
>>> eleves = [ {'INE': 1, 'Nom': 'Nagoumi', 'Prénom': 'Lim'}, \
               {'INE': 2, 'Nom': 'Chagoumi', 'Prénom': 'Zaza'}, \
               {'INE': 3, 'Nom': 'Ghazoumi', 'Prénom': 'Didi'}, \
               {'INE': 4, 'Nom': 'Berroumi', 'Prénom': 'Lili'} ]
>>> assert importTable("élèves.csv") == eleves
```

```
# fichier "INE-classe.csv"
INE;Classe;
1;111;
2;111;
3;122;
4;122;
```

```
>>> INE_classe = [ {'INE': 1, 'Classe': '111'}, \
                   {'INE': 2, 'Classe': '111'}, \
                   {'INE': 3, 'Classe': '122'}, \
                   {'INE': 4, 'Classe': '122'} ]
>>> assert importTable("INE-classe.csv") == INE_classe
```

```
# fichier "INE-moyenne.csv"
INE;Moyenne;
1;13;
2;15;
3;17;
4;19;
```

```
>>> INE_moyenne = [ {'INE': 1, 'Moyenne': 13}, \
                    {'INE': 2, 'Moyenne': 15}, \
                    {'INE': 3, 'Moyenne': 17}, \
                    {'INE': 4, 'Moyenne': 19} ]
>>> assert importTable("INE-moyenne.csv") == INE_moyenne
```

QUESTION 2

Écrire une fonction ***rechercheEleveINE***, qui prend en paramètres une table de données ***table*** et un identifiant ***INE***, et qui renvoie le dictionnaire éventuel correspondant à l'élève de ***table*** qui a ***INE*** pour identifiant :

```
>>> assert rechercheEleveINE(eleves, 1) == {'INE': 1, 'Nom': 'Nagoumi', 'Prénom': 'Lim'}
>>> assert rechercheEleveINE(INE_classe, 1) == {'INE': 1, 'Classe': '111'}
>>> assert rechercheEleveINE(INE_moyenne, 4) == {'INE': 4, 'Moyenne': 19}
```

QUESTION 3

Écrire une fonction **rechercheElevesClasse**, qui prend en paramètres une table de données avec les noms des élèves **eleves**, une table de données avec les classes des élèves **INE_classe** et une classe **classe**, et qui renvoie une table fusionnée avec tous les élèves de la classe :

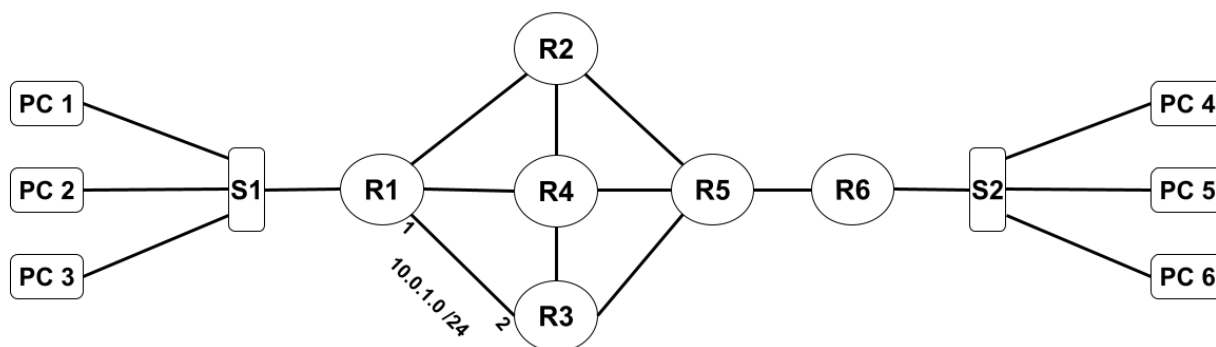
```
>>> table_1 = [{'INE': 1, 'Nom': 'Nagoumi', 'Prénom': 'Lim', 'classe': '111'}, \
               {'INE': 2, 'Nom': 'Chagoumi', 'Prénom': 'Zaza', 'classe': '111'}]
>>> assert rechercheElevesClasse(eleves, INE_classe, '111') == table_1
>>> assert rechercheElevesClasse(eleves, INE_classe, '100') == [ ]
```

QUESTION 4

Programmer une instruction qui construit par compréhension une table contenant tous les élèves d'une certaine classe qui ont une note supérieure ou égale à une certaine note (compléter la ligne 2 sur votre copie) :

```
1 classe = '112'; note = 17
2 table = ..... # à compléter
3 assert table == [{'INE': 3, 'Nom': 'Ghazoumi', 'Prénom': 'Didi', 'Classe': '112', 'Note': 17}, \
4               {'INE': 4, 'Nom': 'Berroumi', 'Prénom': 'Lili', 'Classe': '112', 'Note': 19}]
```

QUESTION 5



- Recopier le réseau ci-dessus et compléter chaque liaison par une adresse réseau et chaque interface de routeur par la partie machine de son adresse, comme pour la liaison R1-R3 où :
 - l'adresse réseau est 10.0.1.0;
 - le masque réseau est à 1 sur les 24 premiers bits et à 0 les 8 derniers bits, donc les trois premiers octets représentent la partie réseau (10.0.1) et le dernier octet représente la partie machine (0); cela signifie donc que le masque vaut 255.255.255.0.
 - la partie machine de l'adresse de l'interface de R1 est 1;
 - la partie machine de l'adresse de l'interface de R3 est 2.
- Sur la base de votre adressage :
 - donner une adresse **IP** à chacun des 6 PC;
 - le résultat d'un **traceroute** depuis le **PC1** vers le **PC4**.

Exercice bonus (optionnel)

Programmer une instruction qui fusionne une liste de tables de données **tables** sur la base d'un identifiant commun qui se trouve dans le premier champ de chaque table, en ne retenant que les lignes communes.