Exercice 1 : Réseaux, routeur et protocole de routage

1. La commande ping renvoie « Hôte inaccessible », cela signifie que l'ordinateur M1 ne réussit pas à établir une route vers l'ordinateur M3. En effet, ces deux machines n'appartiennent pas au même sous-réseau 192.168.1.0 pour M1 et 192.168.2.0 pour M3 et ne peuvent pas donc pas communiquer directement.

2. RAM signifie Random Access Memory.

3. Linux est le nom du noyau d'un système d'exploitation libre, écrit par Linus Torvalds en 1991.

4. Le rôle d'un routeur est d'orienter des messages entre différents réseaux. S'il n'est connecté qu'à un seul réseau, il n'a aucun intérêt. Deux réseaux connectés par deux interfaces du routeur est un minimum.

5. L'interface eth0 du routeur R1 est sur le réseau 192.168.1.0/24. On peut lui attribuer toute adresse de la plage 192.168.1.1 à 192.168.1.254. L'adresse 192.168.1.255 est réservée pour le broadcast (diffusion à tous). On utilise en général la dernière adresse 192.168.1.254 pour le routeur jouant le rôle de passerelle pour un réseau.

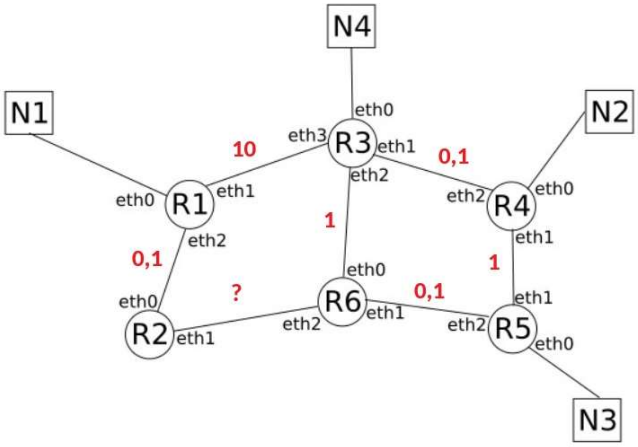
6. Le chemin est N1-R1-R3-R4-N2.

7. N4 n'est plus accessible puisque R3 est en panne.

Table de routage de R1 suite à la panne de R3		
destination	interface de sortie	métrique
N1	eth0	0
N2	eth2	4
N3	eth2	3

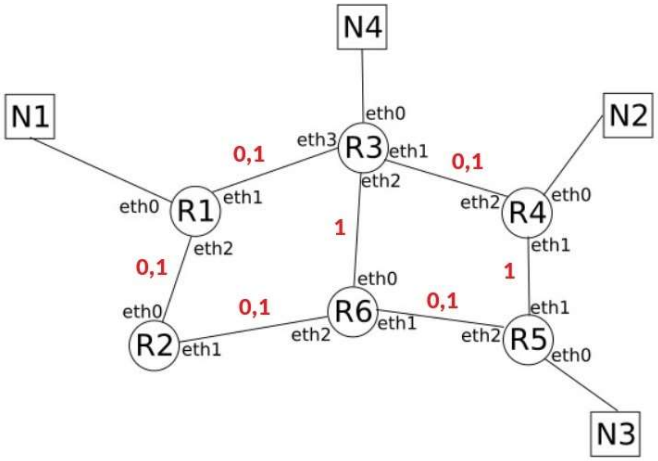
8. Coût d'une liaison fibre : $10^8/10^9 = 0,1$ (1 Gbit/s = 10^9 bit/s)
Coût d'une liaison Fast-Ethernet : $10^8/10^8 = 1$ (100 Mbit/s = 10^8 bit/s)
Coût d'une liaison Ethernet : $10^8/10^7 = 10$ (10 Mbit/s = 10^7 bit/s)

9. La liaison R2 – R6 est une liaison fibre car la métrique pour atteindre N3 depuis R1 par l'interface eth2 vaut 0,3.

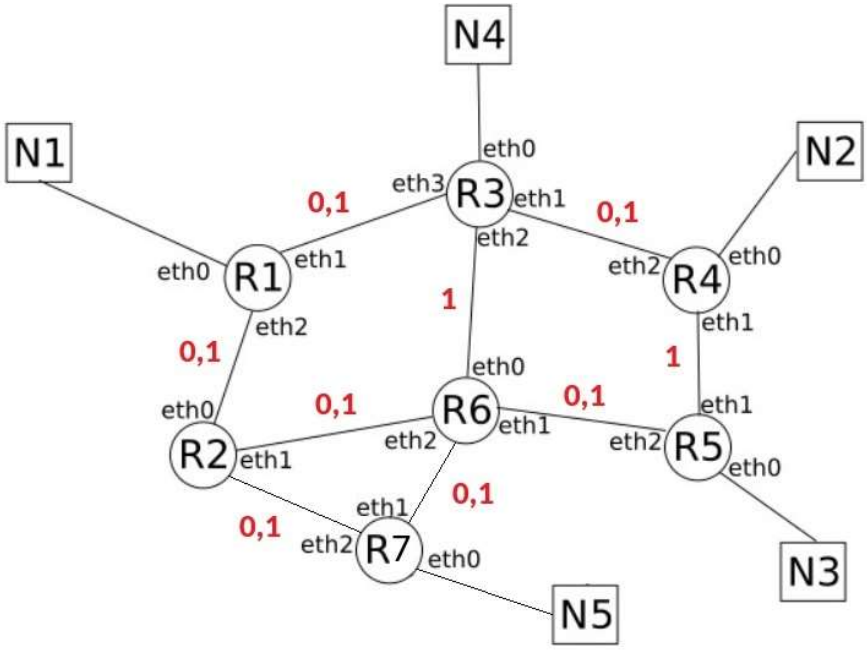


10.

Table de routage de R1		
destination	interface	métrique
N1	eth0	0
N2	eth1	**0,2**
N2	eth2	1,3
N3	eth1	**1,2**
N3	eth2	0,3
N4	eth1	**0,1**
N4	eth2	1,2



11. D'après la table de routage de R1, N5 est atteignable par eth2 avec une métrique de 0,2 donc R7 est lié à R2, et d'après la table de routage de R3, N5 est atteignable par eth2 avec une métrique de 1,1 donc R7 est lié à R6.
Les interfaces eth2 et eth1 de R7 peuvent être interchangée car on ne dispose pas d'assez d'informations pour les positionner.



Exercice 2 : Institut d'enseignement Néo-moderne (EN)

1.
```
def corrige(cop, corr):
    rep = []
    for i, r in enumerate(cop):
        rep.append(r == corr[i])
    return rep
```

Autre solution :
```
def corrige(cop, corr):
    return [cop[i]==corr[i] for i in range(20)]
```

2.
```
def note(cop, corr):
    note = 0
    for i in range(20):
        if cop[i] == corr[i]:
            note += 1
    return note
```

Autre solution :
```
def note(cop, corr):
    note = 0
    for i, r in enumerate(cop):
        if r == corr[i]:
            note += 1
    return note
```

3.
```
def notes_paquet(p, corr):
    rep = {}
    for cle, val in p.items():
        rep[cle] = note(val, corr)
    return rep
```

Autre solution :
```
def notes_paquet(p, corr):
    return {nom:note(cop, corr) for nom, cop in p.items()}
```

4. On ne peut pas utiliser une liste comme clé d'un dictionnaire car les listes sont mutable.

5. Il faudrait associer à chaque candidats un numéro unique d'identification.
Une autre possibilité moins sûre serait d'utiliser comme identifiant un triplet (Prénom, Nom, Date de naissance).

6. Une table d'exécution permet de répondre :

ligne	a	b	c	d	nom	tmp
6	None	None	None	{}		
8	None	None	None	{}	('Tom','Matt')	None
8	(('Tom','Matt'):6)	None	None	{}	('Lambert','Ginne')	None
8	(('Tom','Matt'):6)	(('Lambert','Ginne'),4)	None	{}	('Carl', 'Roth')	None
8	(('Tom','Matt'):6)	(('Lambert','Ginne'),4)	(('Carl','Roth'):2)	{}	('Kurt', 'Jett')	(('Carl','Roth'):2)
8	(('Tom','Matt'):6)	(('Lambert','Ginne'),4)	(('Kurt','Jett'),4)	{('Carl','Roth'):2}	('Ayet', 'Finzerb')	(('Kurt','Jett'),4)
21	(('Tom','Matt'):6)	(('Lambert','Ginne'),4)	(('Kurt','Jett'),4)	{('Carl','Roth'):2,('Ayet','Finzerb'):3}	('Ayet', 'Finzerb')	(('Kurt','Jett'),4)

La valeur renvoyée est `(a, b, c, d)` soit :
((('Tom','Matt'):6), (('Lambert','Ginne'),4), (('Kurt','Jett'),4), {('Carl','Roth'):2, ('Ayet','Finzerb'):3})

7. La fonction `enigme` permet de déterminer le podium des trois meilleurs candidats.

8. Si le dictionnaire contient strictement moins de 3 candidats, ils sont renvoyés dans l'ordre décroissant des notes, avec un ou plusieurs `None` pour compléter le podium, et un dictionnaire vide en quatrième élément du tuple.

9.
```
def classement(notes):
    classement = []
    while notes != {}:
        a, b, c, notes = enigme(notes)
        for e in [a, b, c]:
            if e:
                classement.append(e)
    return classement
```

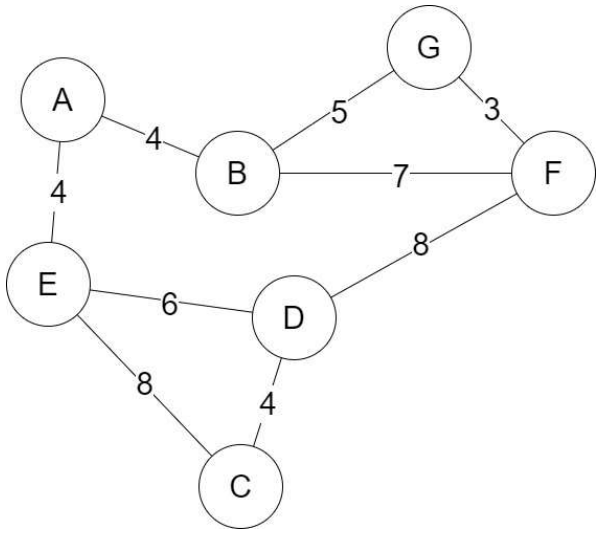
10. Utilisation de la recherche par dichotomie.
```
def renote_express2(copcorr):
    gauche = 0
    droite = len(copcorr)
    while droite - gauche > 1:
        milieu = (gauche + droite)//2
        if copcorr[milieu]:
            gauche = milieu
        else:
            droite = milieu
    if copcorr[gauche]:
        return gauche + 1
    else:
        return gauche
```

11. Le coût en temps de la fonction `renote_express` est linéaire (« en O(n) ») puisque le tableau est entièrement parcouru.
Le coût en temps de la fonction `renote_express2` qui applique le principe de la recherche par dichotomie est logarithmique (« en O (log(n)) »), À chaque étape, le nombre d'éléments du tableau considéré est divisé par deux.

12. Il suffit que la fonction `renote_express2` prenne en paramètre la copie `cop` et la correction `corr` et remplacer `copcorr[milieu]` et `copcorr[gauche]` par respectivement `cop[milieu] == corr[milieu]` et `cop[gauche] == corr[gauche]`.

Exercice 3 : Société CarteMap

1. Graphe pondéré G1 :



2. Le chemin le plus court entre A et D est A-E-D pour une distance de 10.

3. Matrice d'adjacence de G1 :

	A	B	C	D	E	F	G
A	0	4	-	-	4	-	-
B	4	0	-	-	-	7	3
C	-	-	0	4	8	-	-
D	-	-	4	0	6	8	-
E	4	-	9	6	0	-	-
F	-	7	-	8	-	0	3
G	-	3	-	-	-	3	0

4. Exemple de représentation du graphe G2 en python :

```
g2 = {'A':['B', 'C', 'H'],
      'B':['A', 'I'],
      'C':['A', 'D', 'E'],
      'D':['C', 'E'],
      'E':['C', 'D', 'G'],
      'F':['G', 'I'],
      'G':['E', 'F', 'H'],
      'H':['A', 'G', 'I'],
      'I':['B', 'H', 'F']}
```

5. Exemple d'un parcours en largeur du graphe G2 issu de A :
A-B-C-H-D-E-I-G-F

6. La fonction `cherche_itinéraires` est récursive car elle s'appelle elle-même (ligne 8).

7. La fonction `cherche_itinéraires` permet de déterminer tous les chemins possibles pour se rendre d'un point à un autre par une recherche en profondeur. Les chemins sont stockées dans la variable `tab_itineraires`.

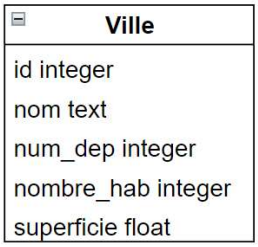
8. La fonction `itineraires_court` permet de sélectionner les plus courts des chemins obtenus par la fonction `cherche_itinéraires`.

```
def itineraires_court(G, dep, arr):
    cherche_itineraires(G, dep, arr)
    tab_court = []
    mini = float('inf')
    for v in tab_itineraires:
        if len(v) <= mini:
            mini = len(v)
    for v in tab_itineraires:
        if len(v) == mini:
            tab_court.append(v)
    return tab_court
```

9. Lors du deuxième appel de `itineraires_court`, la variable `tab_itineraires` n'a pas été réinitialisée. Ainsi elle contient toujours le chemin ['A', 'C', 'E'] du précédent appel. Lors du deuxième appel, tous les autres chemins ajoutés à `tab_itineraires` sont plus longs. Ainsi, c'est le chemin ['A', 'C', 'E'] qui demeure le plus court et qui est le seul à être retourner par le deuxième appel de `itineraires_court`.

10. Le choix d'utiliser un SGBD permet d'éviter la redondance des données et d'assurer leur cohérence.

11. Le schéma relationnel de la table **ville** est :
ville(**id** integer, **nom** text, **num_dep** integer, **nombre_hab** integer, **superficie** float)
ou sous la forme d'une table :



12. L'attribut **id_ville** sert de clé étrangère dans la table **sport** en pointant vers l'attribut **id** (clé primaire) de la table **ville**.

13. Le résultat de la requête SQL est : `[{'nom':'Chamonix'}]`

14. `SELECT nom FROM sport WHERE type = 'piscine';`

15. `UPDATE sport SET note = 7 WHERE id = 3;`
ou :
`UPDATE sport SET note = 7 WHERE nom = 'Ballons perdus';`

16. `INSERT INTO ville VALUES(8, 'Toulouse', 31, 471941, 118);`
ou :
`INSERT INTO ville (id, nom, num_dep, nombre_hab, superficie)`
`VALUES (8, 'Toulouse', 31, 471941, 118);`

17.
`SELECT sport.nom FROM sport JOIN ville ON sport.id_ville=ville.id`
`WHERE ville.nom='Annecy' AND sport.type='mur d'escalade';`