Proposition de correction

Exercice 1

Partie A

Q1.a

192.168.5.254

Q1.b

256 – 2 – 1 (routeur) = 253

Q2.a

255.255.240.0 = 11111111.11111111.11110000.00000000

Q2.b

@IP	192.168.2.2	11000000.10101000.00000010.00000010
masque	255.255.240.0	11111111.11111111.11110000.00000000
@réseau	192.168.0.0	11111111.11111111.11110000.00000000

Q3.c

Permet une continuité de service en cas de panne d'un équipement

Partie B

Q1.a

A → B → E

F → D → A → B

F → D → G → B

F → H → G → B

F → H → E → B

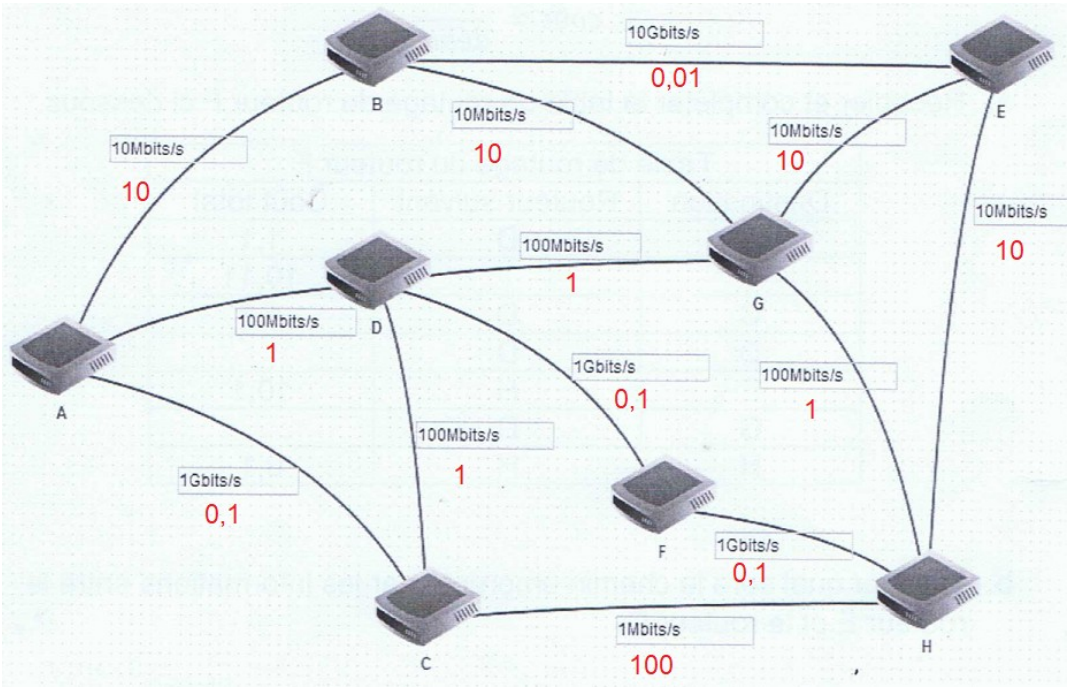
Q1.b

Table de routage du routeur E		
Destination	Routeur suivant	Distance
A	B	2
B	B	1
C	H	2
D	G	2
F	H	2
G	G	1
H	H	1

Table de routage du routeur G		
Destination	Routeur suivant	Distance
A	D	2
B	B	1
C	D	2
D	D	1
E	E	1
F	D	2
H	H	1

Q2.a

Table de routage du routeur F		
Destination	Routeur suivant	Coût total
A	D	1,1
B	H	10,11
C	D	1,1
D	D	0,1
E	H	10,1
G	D	1,1
H	H	0,1



Q2.b

E → H → F → D

Exercice 2

Q1.a

6, 1.70, 100

Q1.b

SELECT nom, age

FROM animal

WHERE nom_espece = ‘bonobo’

ORDER BY age

Q2.a

- clé primaire : nom_espece, identifie une espèce de façon unique
- clé étrangère : num_enclos, met en relation la table espece avec la clé primaire de la table enclos

Q2.b

animal(id_animal, nom, age, taille, poids, #nom_espece)

enclos(num_enclos, ecosysteme, surface, struct, date_entretien)

espece(nom_espece, classe, alimentation, #num_enclos)

Q3.a

UPDATE espece

SET classe = 'mammifères'

WHERE nom_espece = 'ornithorynque'

Q3.b

INSERT INTO animal

VALUES(179, 'Serge', 0, 0.8, 30, 'lama')

Q4.a

SELECT nom, nom_espece

FROM animal

JOIN espece ON animal.nom_espece = espece.nom_espece

JOIN enclos ON espece.num_enclos = enclos.num_enclos

WHERE enclos.struct = 'vivarium' AND espece.alimentation = 'carnovore'

Q4.b

SELECT COUNT(animal.nom)

FROM animal, espece

WHERE espece.classe = 'oiseaux'

AND animal.nom_espece = espece.nom_espece

Exercice 3

Q1.a

'Bonjour Alan !'

Q1.b

bool

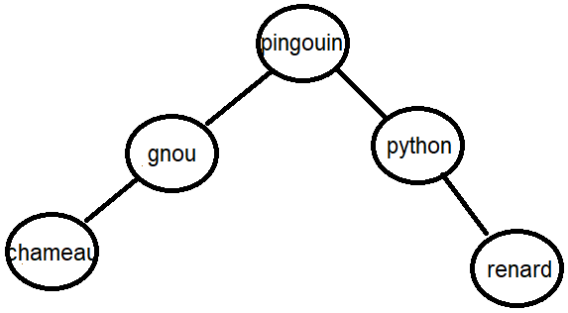
- False (x)

- True (y)

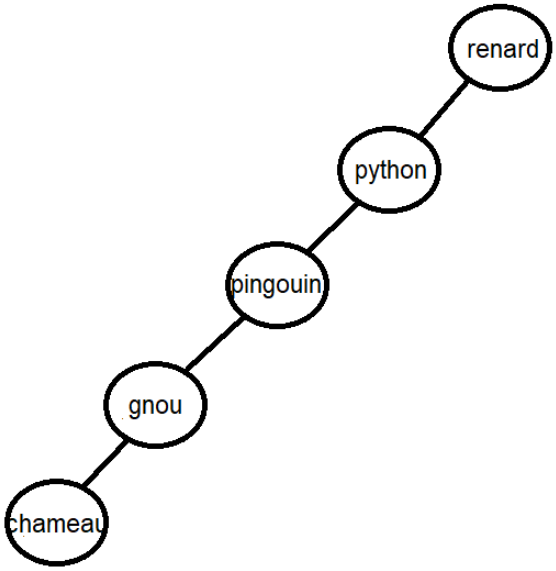
Q1.c

```
def occurrences_lettre(une_chaine : str, une_lettre : str) -> int :
    compteur = 0
    for lettre in une_chaine :
        if lettre == une_lettre :
            compteur += 1
    return compteur
```

Q2.a



Q2.b



Q3.a

calcule la taille de l'arbre, donc 336531

Q3.b

```
def hauteur(un_abr) :
    if un_abr.est_vide() == 0 :
```

```
        return 0
    else :
        return 1 + max(hauteur(un_abr.sous_arbre_gauche), hauteur(un_abr.sous_arbre_droit)
```

Q4.a

```
def chercher_mots(liste_mots, longueur, lettre, position) :
    # assert position < longueur
    res = []
    for i in range(len(liste_mots)) :
        if len(liste_mots[i]) == longueur and liste_mots[i][position]) == lettre :
            res.append(liste_mots[i])
    return res
```

Q4.b

donne les mots de 3 lettres se terminant par ‘ax’

Q4.c

chercher_mots(chercher_mots(chercher_mots(liste_mots_francais, 5, ‘r’, 4), 5, ‘e’, 3), 5, ‘t’, 2)