Autotest n°1 – Correction

Exercice 1 : *Cet exercice est consacré à l'analyse et à l'écriture de programmes récursifs.*

1) a) Expliquer en quelques mots ce qu'est une fonction récursive.
Solution : Une fonction récursive est une fonction qui s'appelle elle-même dans son code.

b) On considère la fonction Python suivante :

```
def compte_rebours(n):
    """ n est un entier positif ou nul """
    if n >= 0:
        print(n)
        compte_rebours(n-1)
```

L'appel `compte_rebours(3)` affiche successivement les nombres 3, 2, 1 et 0.
Expliquer pourquoi le programme s'arrête après l'affichage du nombre 0.
Solution : Après l'affichage de 0, `n` vaut -1. La condition du `if` n'est pas vérifiée et l'exécution de la fonction se termine.

2) En mathématiques, la **factorielle** d'un entier naturel n est le produit des nombres entiers strictement positifs inférieurs ou égaux à n. Par convention, la factorielle de 0 est 1. Par exemple :

- la factoriel de 1 est 1
- la factoriel de 2 est $2 \times 1 = 2$
- la factoriel de 3 est $3 \times 2 \times 1 = 6$
- la factoriel de 4 est $4 \times 3 \times 2 \times 1 = 24$

Compléter le programme ci-dessous afin que la fonction récursive `fact` renvoie la factorielle de l'entier passé en paramètre de cette fonction.
Exemple : `fact(4)` renvoie 24.

```
def fact(n):
    """ Renvoie le produit des nombres entiers
        strictement positifs inférieurs à n """
    if n == 0:
        return 1
    else:
        return n * fact(n-1)
```

3) La fonction `somme_entiers_rec` ci-dessous permet de calculer la somme des entiers, de 0 à l'entier naturel `n` passé en paramètre. Par exemple :

- Pour `n` = 0, la fonction renvoie la valeur 0.
- Pour `n` = 1, la fonction renvoie la valeur $0 + 1 = 1$.

...

- Pour `n` = 4, la fonction renvoie la valeur $0 + 1 + 2 + 3 + 4 = 10$.

```
1 def somme_entier_rec(n):
2     """ Permet de calculer la somme des entiers,
3     de 0 à l'entier naturel n """
4     if n == 0:
5         return 0
```

```
6       else:
7           print(n)   # pour vérifier
8           return n + somme_entiers_rec(n-1)
```

L'instruction `print(n)` de la ligne 7 dans le code précédent a été insérée afin de mettre en évidence le mécanisme en œuvre au niveau des appels récursifs.

a) Écrire ce qui sera affiché dans la console après l'exécution de la ligne suivante :
`res = somme_entiers_rec(3)`
Solution : Il va s'afficher 3, 2 et 1.

b) Quelle valeur sera alors affectée à la variable `res` ?
Solution : La valeur renvoyée est $3+2+1+0=6$.

4) Écrire en Python une fonction `somme_entiers` non récursive : cette fonction devra prendre en argument un entier naturel `n` et renvoyer la somme des entiers de 0 à `n` compris. Elle devra donc renvoyer le même résultat que la fonction `somme_entiers_rec` définie à la question 3. Exemple : `somme_entiers(4)` renvoie 10.

```
def somme_entier(n):
    s = n
    for i in range(n):
        s = s + i
    return s
```

Exercice 2 : *Cet exercice porte sur les langages et la programmation (récursivité).*

```
def f(n):
    if n == 0:
        print("Partez !")
    else:
        print(n)
        f(n-1)
```

1) Voici une fonction codée en Python :

a) Qu'affiche la commande `f(5)` ?
Solution : On va obtenir 5, 4, 3, 2, 1 et `Partez !`.

b) Pourquoi dit-on de cette fonction qu'elle est récursive ?
Solution : La fonction s'appelle elle-même. Elle est donc récursive.

2) On rappelle qu'en Python l'opérateur + a le comportement suivant sur les chaines de caractères et les listes :

```
>>> S = 'a' + 'bc'
>>> S
'abc'
```

```
>>> L = ['a'] + ['b', 'c']
>>> L
['a', 'b', 'c']
```

On a besoin pour les questions suivantes de pouvoir ajouter une chaine de caractères `s` en préfixe à chaque chaine de caractères de la liste `chaines`.
On appellera cette fonction `ajouter`.
Par exemple, `ajouter("a", ["b", "c"])` doit renvoyer `["ab", "ac"]`.

a) Compléter le code suivant :

```
def ajouter(s, chaines):
    resultat = []
    for mot in chaines:
        resultat.append(s + mot)
    return resultat
```

b) Que renvoie la commande `ajouter("b", ["a", "b", "c"])` ?
Solution : `["ba", "bb", "bc"]`.

c) Que renvoie la commande `ajouter("a", [""])` ?
Solution : `["a"]`.

3) On s'intéresse ici à la fonction suivante écrite en Python où `s` est une chaine de caractères et `n` un entier naturel.

```
def produit(s, n):
    if n == 0:
        return [""]
    else:
        resultat = []
        for c in s:
            resultat = resultat + ajouter(c, produit(s, n - 1))
        return resultat
```

a) Que renvoie la commande `produit("ab", 0)` ? Le résultat est-il une liste vide ?
Solution : On obtient `[""]`. Ce n'est pas une liste vide, puisqu'elle contient le texte vide.

b) Que renvoie la commande `produit("ab", 1)` ?
Solution : `["a", "b"]`.

c) Que renvoie la commande `produit("ab", 2)` ?
Solution : `["aa", "ab", "ba", "bb"]`.

Exercice 3 : *Cet exercice porte sur l'algorithmique et la programmation.*
Un **palindrome** est un mot qui se lit de la même manière de la gauche vers la droite que de la droite vers la gauche. Par exemple, *kayak* est un palindrome.
On propose ci-dessous une fonction pour tester si un mot est un palindrome.
On précise que, pour une chaîne de caractères `chaine` :

- l'instruction `len(chaine)` renvoie sa longueur ;
- l'instruction `chaine[-1]` renvoie son dernier caractère ;
- l'instruction `chaine[1:-1]` renvoie la chaîne privée de son premier caractère et de son dernier caractère.

```
def tester_palindrome(chaine):
    if len(chaine) < 2:
        return True
    elif chaine[0] != chaine[-1]:
        return False
    else:
        chaine = chaine[1:-1]
        return tester_palindrome(chaine)
```

1) On saisit, dans la console, l'instruction suivante :

```
>>> tester_palindrome('kayak')
```

Combien de fois est appelée la fonction `tester_palindrome` lors de l'exécution de cette instruction ? On veillera à compter l'appel initial.
Solution : L'appel `tester_palindrome('kayak')` va provoquer l'appel `tester_palindrome('aya')` qui va provoquer l'appel `tester_palindrome('y')`. Il y aura donc 3 appels.

2) a) Justifier que la fonction `tester_palindrome` est récursive.
Solution : Elle est récursive parce qu'elle s'appelle elle-même.

b) Expliquer pourquoi l'appel à la fonction `tester_palindrome` se terminera quelle que soit la chaîne de caractères sur laquelle elle s'applique.

Solution : Chaque nouvel appel récursif se fait avec une chaîne de taille diminuée de 2. Dans le "pire" des cas, on arrivera à un appel récursif avec une chaîne vide ou de taille 1, ce qui arrête les appels récursifs.

3) La saisie, dans la console, de l'instruction `tester_palindrome(53235)` génère une erreur.

a) Parmi les quatre propositions suivantes, indiquer le type d'erreur affiché :

- `ZeroDivisionError`
- `ValueError`
- `TypeError` C'est parce que la fonction va vouloir utiliser `len(53235)`, ce qui provoque l'erreur puisque la fonction `len` ne peut pas s'appliquer sur un entier.
- `IndexError`

b) Proposer sur la copie une ou plusieurs instructions qu'on pourrait écrire en première ligne de la fonction `tester_palindrome` et permettant d'afficher clairement cette erreur à l'utilisateur.

Solution : On peut rajouter `assert type(chaine) == str, "Il faut donner un texte"`.

4) Écrire le code d'une fonction itérative (non récursive) `est_palindrome` qui prend en paramètre une chaîne de caractères et renvoie un booléen égal à `True` si la chaîne de caractères est un palindrome, `False` sinon.

Solution :

```
def est_palindrome(chaine):
    n = len(chaine)
    for i in range(n//2):
        if chaine[i] != chaine[n-i-1]:
            return False
    return True
```

Exercice 4 : *Cet exercice porte sur les réseaux et les protocoles de routages.*

Quelques rappels :
Une adresse IPv4 est composée de 4 octets `X1.X2.X3.X4` qui peuvent être écrits en notation binaire ou décimale.
La notation CIDR `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 la partie « hôte » (machine).

1) a) Donner le nombre de bits formant un octet.

Solution : Il faut 8 bits pour faire un octet.

b) Déterminer l'écriture décimale de l'adresse IPv4 correspondant à l'écriture binaire :

`11000000.10101000.00000100.11110001`

Solution :

128	64	32	16	8	4	2	1	
1	1	0	0	0	0	0	0	192
1	0	1	0	1	0	0	0	168
0	0	0	0	0	1	0	0	4
1	1	1	1	0	0	0	1	241

L'adresse est donc `192.168.4.241`.

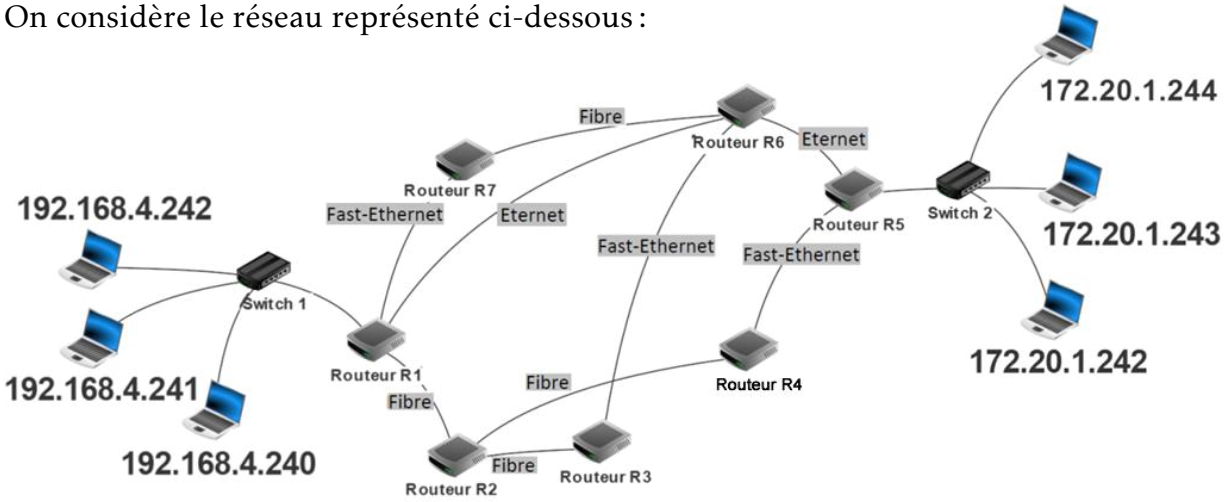
2) On considère la machine d'adresse IPv4 `172.20.1.242/24`.

a) Donner la notation décimale du masque de sous-réseau de cette machine.
Solution : La notation binaire est `11111111.11111111.11111111.00000000`, ce qui correspond à `255.255.255.0`.

b) Donner l'adresse décimale de ce réseau.
Solution : L'adresse du réseau est `172.20.1.0`.

c) Donner le nombre maximal de machines que l'on peut connecter sur ce réseau.
Solution : Il y a $2^8 = 256$ adresses possibles. En enlevant l'adresse du réseau et l'adresse de broadcast (`172.20.1.255`), il reste donc 254 adresses disponibles pour des machines.

3) On considère le réseau représenté ci-dessous :



Le protocole RIP (Routing Information Protocol) est un protocole de routage qui cherche à minimiser le nombre de routeurs traversés (ce qui correspond à la distance ou au nombre de sauts).
Le réseau est composé de 7 routeurs : R1, R2, R3, R4, R5, R6 et R7 et utilise le protocole RIP. Le routeur R1 doit transmettre des données au routeur R5.

a) Déterminer le parcours pouvant être emprunté par ces données en vous aidant des tables de routage ci-dessous.
Solution : Le parcours est : R1–R6–R5.

Table de R1	
Dest.	Suiv.
R2	R2
R3	R6
R4	R2
R5	R6
R6	R6
R7	R7

Table de R2	
Dest.	Suiv.
R1	R1
R3	R3
R4	R4
R5	R4
R6	R3
R7	R1

Table de R3	
Dest.	Suiv.
R1	R6
R2	R2
R4	R2
R5	R6
R6	R6
R7	R6

Table de R4	
Dest.	Suiv.
R1	R2
R2	R2
R3	R2
R5	R5
R6	R5
R7	R2

Table de R5	
Dest.	Suiv.
R1	R6
R2	R4
R3	R6
R4	R4
R6	R6
R7	R6

Table de R6	
Dest.	Suiv.
R1	R1
R2	R3
R3	R3
R4	R5
R5	R5
R7	R7

Table de R7	
Dest.	Suiv.
R1	R1
R2	R1
R3	R6
R4	R7
R5	R6
R6	R6

Pour les deux questions suivantes, on suppose que la liaison entre R1 et R6, est coupée.

b) Compléter la nouvelle table de routage de R1 ci-contre.

c) En déduire le parcours que suivront les données pour aller du routeur R1 au routeur R5.

Solution : Il y a deux chemins possibles : R1–R7–R6–R5 et R1–R2–R4–R5.

Table de R1	
Dest.	Suiv.
R2	R2
R3	R2
R4	R2
R5	R2 ou R7
R6	R7
R7	R7

4) Pour la suite de l'exercice, on considère que la liaison entre R1 et R6 a été rétablie et on applique désormais le protocole de routage OSPF attribuant un coût à chaque liaison afin de trouver le chemin permettant une transmission plus rapide.

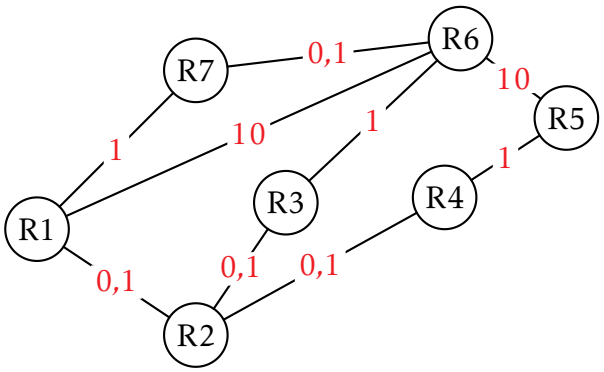
Le coût d'une liaison est défini par la relation :

$$\text{coût} = \frac{10^8}{d} \quad \text{où } d \text{ représente le débit en bit} \cdot \text{s}^{-1}$$

Liaison	Débit	Coût
Ethernet	10^7	10
Fast-Ethernet	10^8	1
Fibre	10^9	0,1

a) Compléter le tableau suivant :

b) Compléter la figure ci-dessous en faisant apparaître le coût de chacune des liaisons.



Le coût d'un chemin est la somme des coûts des liaisons empruntées.

c) Donner les 6 chemins possibles ainsi que leur coût lors de l'envoi d'un paquet depuis le routeur R1 vers le routeur R5.

Solution : Il y a :

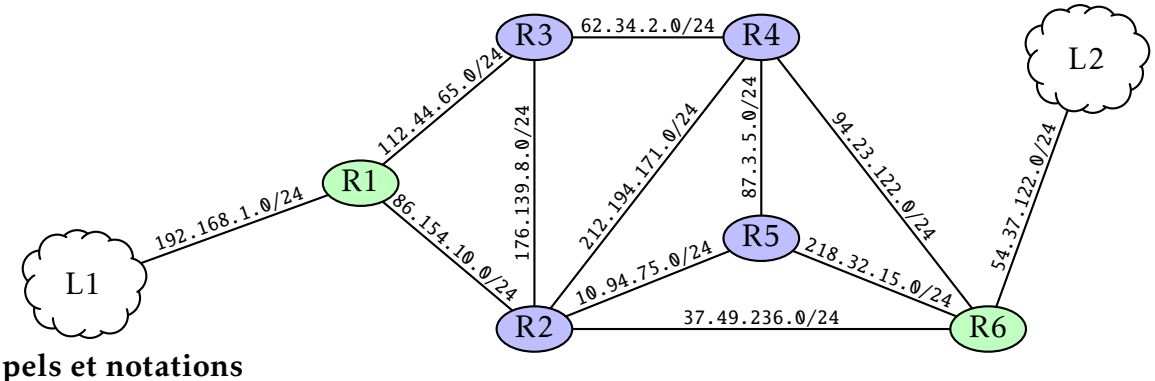
- R1–R7–R6–R5 qui a un coût de 11,1.
- R1–R7–R6–R3–R2–R4–R5 qui a un coût de 3,3.
- R1–R6–R5 qui a un coût de 20.
- R1–R6–R3–R2–R4–R5 qui a un coût de 22,2.
- R1–R2–R4–R5 qui a un coût de 1,2.
- R1–R2–R3–R6–R5 qui a un coût de 11,2.

d) Déduire, en respectant le protocole OSPF, le chemin le moins coûteux lors de l'envoi d'un paquet depuis le routeur R1 vers le routeur R5. Préciser le coût minimal.

Solution : Le chemin sera donc R1–R2–R4–R5 qui a un coût de 1,2.

Exercice 5 : On représente ci-dessous un réseau dans lequel R1, R2, R3, R4, R5 et R6 sont

des routeurs. Le réseau local L1 est relié au routeur R1 et le réseau local L2 au routeur R6.



Rappels et notations

Dans cet exercice, les adresses IP sont composées de 4 octets, soit 32 bits. Elles sont notées X1.X2.X3.X4, où X1, X2, X3 et X4 sont les valeurs des 4 octets, convertis 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 représentent la partie « hôte ».

Toutes les adresses des hôtes connectés à un réseau local ont la même partie réseau et peuvent donc communiquer directement. L'adresse IP dont tous les bits de la partie « hôte » sont à 0 est appelée « adresse du réseau ».

On donne également des extraits de la table de routage des routeurs R1 à R5 dans le tableau suivant :

Routeur	Réseau destinataire	Passerelle	Interface
R1	`54.37.122.0/24`	`86.154.10.1`	`86.154.10.56`
R2	`54.37.122.0/24`	`37.49.236.22`	`37.49.236.23`
R3	`54.37.122.0/24`	`62.34.2.8`	`62.34.2.9`
R4	`54.37.122.0/24`	`94.23.122.10`	`94.23.122.11`
R5	`54.37.122.0/24`	`218.32.15.1`	`218.32.15.2`

1) Un paquet part du réseau local L1 à destination du réseau local L2.
   a) En utilisant l'extrait de la table de routage de R1, vers quel routeur R1 envoie-t-il ce paquet : R2 ou R3 ? Justifier.
   Solution : D'après la ligne de la table du routeur R1, il faut l'envoyer sur le réseau `86.154.10.0/24` qui le relie à R2.
   b) À l'aide des extraits de tables de routage ci-dessus, nommer les routeurs traversés par ce paquet, lorsqu'il va du réseau L1 au réseau L2.
   Solution : Le chemin suivi est R1–R2–R6.

2) La liaison entre R1 et R2 est rompue.
   a) Sachant que ce réseau utilise le protocole RIP (distance en nombre de sauts), donner l'un des deux chemins possibles que pourra suivre un paquet allant de L1 vers L2.
   Solution : Sans la liaison entre R1 et R2, il est possible de suivre R1–R3–R4–R6 ou R1–R3–R2–R6.
   b) Dans les extraits de tables de routage ci-dessus, pour le chemin de la question **2)a)**, quelle(s) ligne(s) sera (seront) modifiée(s) ? Vous indiquerez simplement le(s) routeur(s) dont la table est modifiée et le nouveau routeur qui sert de passerelle.
   Solution : Il n'y a que la table de R1 qui doit être modifiée. Il faut envoyer les paquets à R3.

3) On a rétabli la liaison entre R1 et R2.
   Par ailleurs, pour tenir compte du débit des liaisons, on décide d'utiliser le protocole OSPF (distance liée au coût minimal des liaisons) pour effectuer le routage. Le coût des liaisons entre les routeurs est donné par le tableau suivant :

Liaison	R1-R2	R1-R3	R2-R3	R2-R4	R2-R5	R2-R6	R3-R4	R4-R5	R4-R6	R5-R6
Coût	100	100	?	1	10	10	10	1	10	1

a) Le coût $\mathscr{C}$ d'une laison est donné ici par la formule :

$$\mathscr{C} = \frac{10^9}{\text{BP}}$$

où BP est la bande passante de la connexion en bps (bit par seconde).
Sachant que la bande passante de la liaison R2-R3 est de 10 Mbps, calculer le coût correspondant. On rappelle que $10\,\text{Mbps} = 10^7$ bps.
Solution : On a :

$$\mathscr{C} = \frac{10^9}{10^7} = 100$$

b) Déterminer le chemin parcouru par un paquet partant du réseau L1 et arrivant au réseau L2, en utilisant le protocole OSPF. Vous indiquerez également le coût.
Solution : Le chemin R1–R2–R4–R5–R6 a un coût de 103, qui est le coût minimal.

c) Compléter les extraits des tables de routages à destination de L2, avec la métrique OSPF. Vous utiliserez le nom du routeur qui sert de passerelle plutôt que son adresse.

Routeur	Réseau destinataire	Passerelle	Métrique OSPF
R1	`54.37.122.0/24`	R2	103
R2	`54.37.122.0/24`	R4	3
R3	`54.37.122.0/24`	R4	12
R4	`54.37.122.0/24`	R5	2
R5	`54.37.122.0/24`	R6	1