Lycée Les 3 Sources **Bourg-Lès-Valence**
NSI T^le Année **2024-25**

Devoir surveillé n°1 – Correction

Nom et prénom :

Exercice 1 : (11pt) *Cet exercice porte sur la récursivité.*
Au rugby, une équipe peut marquer, pour simplifier :

- soit 3 points (pénalité) ;
- soit 5 points (essai non transformé) ;
- soit 7 points (essai transformé).

On souhaite savoir s'il est possible d'obtenir un score donné avec uniquement des pénalités, c'est-à-dire avec une succession de "coups" à 3 points.

1) Écrire une fonction `possible_avec_penalites_seules` qui prend en paramètre un score et qui renvoie `True` si le score passé en paramètre peut être marqué uniquement avec des pénalités.

Exemple :

```
>>> possible_avec_penalites_seules(15)
True
>>> possible_avec_penalites_seules(10)
False
```

```
def possible_avec_penalites_seules(score):
    return score % 3 == 0
```

2) Compléter le tableau suivant qui précise, pour les scores de 0 à 10, les évolutions du score menant à un total donné et le nombre de façons différentes d'obtenir ce total. Par exemple, pour obtenir un score de 8, en partant de 0, il y a 2 possibilités :
   - Soit l'équipe marque un essai non transformé, atteignant 5 points, puis une pénalité, atteignant 8 points ;
   - Soit l'équipe marque une pénalité, atteignant 3 points, puis un essai non transformé, atteignant 8 points.

Score	Liste des solutions	nombre de solutions
0	[0]	1
1	[]	0
2	[]	0
3	[0, 3]	1
4	[]	0
5	[0, 5]	1
6	[0, 3, 6]	1
7	[0, 7]	1
8	[0, 5, 8], [0, 3, 8]	2
9	[0, 3, 6, 9]	1
10	[0,3,10],[0,5,5],[0,7,10]	3

En notant $f(n)$ le nombre de possibilités d'obtenir le score n, on admet que pour $n > 6$ on a la relation suivante :

$$f(n) = f(n-3) + f(n-5) + f(n-7)$$

3) Vérifier cette relation pour $n = 10$ à l'aide du tableau établi à la question 2.
Solution : D'après le tableau, on a $f(10) = 3$, $f(7) = 1$, $f(5) = 1$ et $f(3) = 1$).
On a donc $f(10) = f(7) + f(5) + f(3)$.

On veut écrire une fonction récursive `nb_solutions`, qui prend en paramètre un entier positif quelconque correspondant à un score, et qui renvoie le nombre de façons d'obtenir ce score donné.

4) Déterminer le résultat de cette fonction récursive pour tous les cas de base, c'est-à-dire pour chaque entier `n` de 0 à 6.
Solution : Si `n` vaut 0, 3, 5 ou 6, le résultat est 1. Et si `n` vaut 1, 2 ou 4, le résultat est 0.

5) Écrire la fonction récursive `nb_solutions`, qui prend en paramètre un entier positif quelconque correspondant à un score, et qui renvoie le nombre de possibilités d'obtenir ce score donné.

```
def nb_solutions(n):
    if n in [1, 2, 4]:
        return 0
    elif n in [0, 3, 5, 6]:
        return 1
    else:
        return nb_solutions(n-3)+nb_solutions(n-5)+nb_solutions(n-7)
```

6) On se rend compte que pour `n` > 100, l'évalutation de `nb_solutions(n)` devient très très lente. Expliquer brièvement pourquoi.
Solution : À chaque appel, cette fonction fait 3 appels récursifs. Lorsque `n` est grand, cela donne un nombre d'appels récursifs, ce qui est très lent.

On veut écrire une fonction récursive `solutions_possibles`, qui prend en paramètre un entier positif quelconque correspondant à un score, et qui renvoie la liste composée de toutes les listes représentant les possibilités d'obtenir ce score.
Par exemple :

```
>>> solutions_possibles(8)
[[0, 5, 8], [0, 3, 8]]
```

Nom et prénom :

7) a) Déterminer quelles lignes du tableau permettent de construire rapidement la liste renvoyée par l'appel `solutions_possibles(11)`.
Solution : Il faut utiliser les lignes 8, 6 et 4.

b) Déterminer le résultat obtenu pour `solutions_possibles(11)`
Solution : Le résultat sera `[[0, 5, 8, 11], [0, 3, 8, 11], [0, 3, 6, 11]]`

8) Compléter la fonction `solutions_possibles` suivante, qui prend en paramètre un score et qui renvoie la liste des possibilités d'obtenir ce score :

```
def solutions_possibles(score):
    if score < 0:
        resultat = []
    elif score == 0:
        resultat = [[0]]
    else:
        resultat = []
        for coup in [3, 5, 7]:
            liste = solutions_possibles(score-coup)
            for solution in liste:
                solution.append(score)
                resultat.append(solution)
    return resultat
```

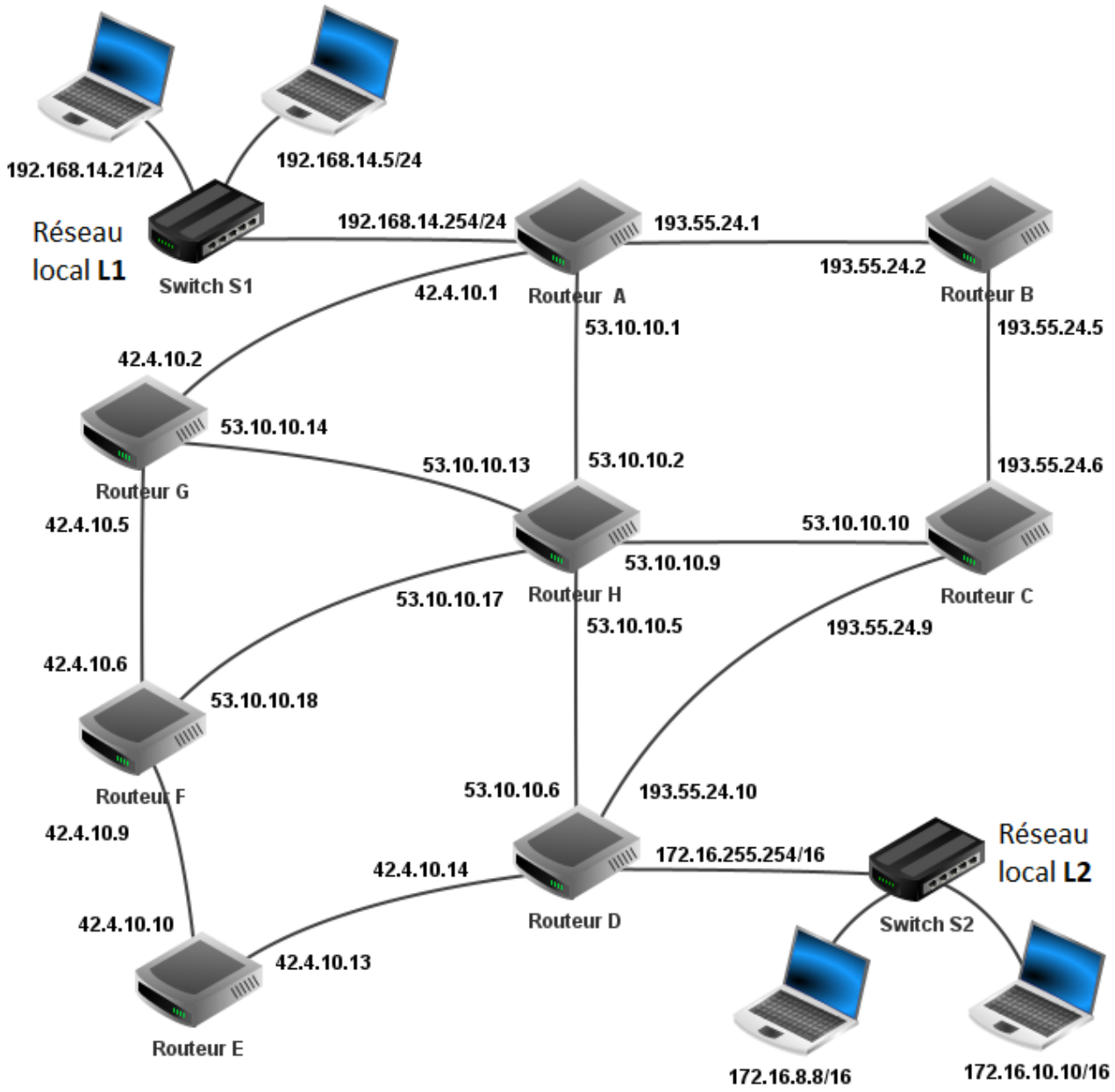
Exercice 2 : (7pt) *Cet exercice porte sur les réseaux et les protocoles de routage.*
Rappels :
Une adresse IPv4 est composée de 4 octets, soit 32 bits. Elle est notée `a.b.c.d`, où `a`, `b`, `c` et `d` sont les valeurs décimales des 4 octets et nommée « notation décimale pointée ».
La notation `a.b.c.d/n`, appelée notation CIDR (Classless Inter Domain Routing), signifie que les n premiers bits à gauche de l'adresse IP représentent la partie « réseau », les bits à droite qui suivent représentent la partie « machine ».
L'adresse IPv4 dont tous les bits de la partie « machine » sont à 0 est appelée « adresse du réseau ».
L'adresse IPv4 dont tous les bits de la partie « machine » sont à 1 est appelée « adresse de diffusion ».
On considère le réseau représenté sur la page suivante.

Partie A : Adresses IP

1) Les machines du réseau local L1 indiquent un masque de sous réseau sur 24 bits en notation CIDR, soit `255.255.255.0` en notation décimale pointée.
Donner le masque de sous réseau en notation décimale pointée des machines du réseau L2 (masque de sous réseau de 16 bits). **Solution :** `255.255.0.0`

Concernant le réseau local L2 :

2) Donner l'adresse du réseau. **Solution :** `172.16.0.0`
3) Donner l'adresse de diffusion. **Solution :** `172.16.255.255`
4) Donner le nombre maximum de machines pouvant être connectées à ce réseau. On pourra donner une expression mathématique sans faire le calcul.
Solution : Il y a $2^{16} - 2 = 65534$ machines qui peuvent être connectées.



Partie B : Protocoles de routage

On donne ci-dessous des extraits des tables de routage des routeurs :

Routeur	Réseau destinataire	Passerelle	Interface
A	L2	53.10.10.2	53.10.10.1
B	L2	193.55.24.6	193.55.24.5
C	L2	193.55.24.10	193.55.24.9
D	L2	Connecté	172.16.255.254
E	L2	42.4.10.14	42.4.10.13
F	L2	42.4.10.10	42.4.10.9
G	L2	53.10.10.13	53.10.10.14
H	L2	53.10.10.6	53.10.10.5

5) À l'aide des extraits des tables de routage ci-dessus, donner un chemin (c'est-à-dire nommer les routeurs traversés) suivi par un message envoyé du réseau L1 vers le réseau L2.

Solution : L1–A–H–D–L2

Nom et prénom :

La liaison entre les routeurs H et D est rompue :

6) Sachant que le protocole de routage RIP est utilisé (distance en nombre de sauts), donner les nouveaux chemins que pourra suivre un message allant de L1 vers L2.
Solution : L1–A–H–C–D–L2 ou L1–A–B–C–D–L2

7) Choisir un des chemins de la question précédente. Indiquer le ou les routeurs dont la règle à destination de L2 doit être modifiée pour obtenir ce chemin. Préciser la nouvelle règle de routage pour ce(s) routeur(s).
Solution : Pour, L1–A–H–C–D–L2 il faut changer la ligne du H avec `53.10.10.9` comme passerelle et `53.10.10.10` comme interface.
Pour L1–A–B–C–D–L2 il faut changer la ligne du A avec `193.55.24.2` comme passerelle et `193.55.24.1` comme interface.

La liaison entre les routeurs H et D est rétablie.
Pour tenir compte du débit des liaisons, on décide d'utiliser le protocole OSPF (distance liée au coût des liaisons) pour effectuer le routage.
Le coût d'une liaison est donné ici par la formule :

$$\text{coût} = \frac{10^9}{\text{BP}} \quad \text{où BP est la bande passante de la connexion en bit par seconde.}$$

Voici les valeurs des bandes passantes de chaque liaison entre les routeurs :

Liaison	Bande passante
A-B	1 Gbit/s
A-H	1 Gbit/s
A-G	1 Gbit/s
B-C	1 Gbit/s
C-H	100 Mbit/s
C-D	1 Gbit/s

Liaison	Bande passante
D-H	100 Mbit/s
D-E	10 Gbit/s
E-F	10 Gbit/s
F-H	1 Gbit/s
F-G	10 Gbit/s
G-H	1 Gbit/s

8) Calculer le coût des liaisons pour les 3 valeurs de bande passante qui apparaissent dans le tableau ci-dessus.
Solution : On a :
- Pour 10 Gbit/s, $c = \frac{10^9}{10^{10}} = 0{,}1$.
- Pour 1 Gbit/s, $c = \frac{10^9}{10^9} = 1$.
- Pour 100 Mbit/s, $c = \frac{10^9}{10^8} = 10$.

9) Déterminer alors le chemin que suivra un message allant de L1 vers L2 et donner son coût.
Solution : L1–A–G–F–E–D–L2, avec un coût de 1,3.

10) La liaison entre les routeurs G et F est rompue. Déterminer le nouveau chemin suivi par un message allant de L1 vers L2 et donner son coût.
Solution : L1–A–H–F—-E–D–L2, avec un coût de 2,2.