EXERCICE 2 (6 points)

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).

Partie A

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
```

2. Recopier et 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		
3		
4		
5		
6		
7		
8	[0, 5, 8], [0, 3, 8]	2
9		1
10	[0,3,10]	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.

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 tous les cas de base, pour chaque entier n de 0 à 6, de cette fonction récursive.

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.

6. Lors de l'appel de `nb_solutions(score)`, on se rend compte que le nombre d'appels récursif augmente très rapidement lorsque `score` augmente. Nommer une méthode algorithmique optimisant le nombre d'appels récursifs.

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]]
```

7. Déterminer quelles lignes du tableau permettent de construire rapidement la liste renvoyée par l'appel `solutions_possibles(11)`.

8. Recopier et 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 = ...
    else:
        resultat = []
        for coup in [..., 5, ...]:
            liste = solutions_possibles(score - coup)
            for solution in liste:
                solution.append(...)
            resultat.append(...)
    return resultat
```