```
"""
Author: Pascal Padilla
Source: correction de l'exercice 2 du sujet 25 des épreuves pratiques NSI 2022

Remarque:
    algorithme qui fonctionne sur diviser pour régner
    rapide car la taille de la zone à explorer est divisée par deux à chaque appel
    ressemble à la dichotomie
"""


def trouver_intrus(tab, g, d):
    '''
    Renvoie la valeur de l'intrus situé entre les indices g et d
    dans la liste tab où
    tab vérifie les conditions de l'exercice,
        g et d sont des multiples de 3.
    '''
    # affichage à décommenter pour 'voir' l'algo fonctionner
    # print(f'recherche: {g}..{d}\n  sous-tableau: {tab[g:d+1]}')

    # cas où la zone de recherche est ramenée à un seul élément
    #  -> l'intrus est identifié
    if g == d:
        return tab[g] # ou (c'est équivalent): return tab[d]

    else:
        # il y a autant de triplets que la longueur de la zone de
        # recherche divisée par 3
        nombre_de_triplets = (d - g) // 3
        # positionnement à l'indice situé au milieu ET
        #  correspondant au début d'un triplet
        indice = g + 3 * (nombre_de_triplets // 2)

        # cas de la zone inférieure à l'intrus
        # (l'élément courant et l'élément juste à droite sont égaux)
        if tab[indice] == tab[indice + 1] :
            # appel récursif avec recherche dans la zone après indice
            #  comme tab[indice] n'est pas l'intrus, on l'exclu de la
            #  zone de recherche à venir: g <- indice + 3
            return trouver_intrus(tab, indice + 3, d)
        else:
            # appel récursif avec recherche dans la zone avant indice
            #  comme tab[indice] est PEUT ÊTRE l'intrus, on NE l'exclu PAS
            #  de la zone de recherche à venir: d <- indice
            return trouver_intrus(tab, g, indice)


# ajout de cette condition pour des raisons de tests automatisés
if __name__ == '__main__':
    assert trouver_intrus([3, 3, 3, 9, 9, 9, 1, 1, 1, 7, 2, 2, 2, 4, 4, 4, 8, 8, 8,
5, 5, 5], 0, 21) == 7
    assert trouver_intrus([8, 5, 5, 5, 9, 9, 9, 18, 18, 18, 3, 3, 3], 0, 12) == 8
    assert trouver_intrus([5, 5, 5, 1, 1, 1, 0, 0, 0, 6, 6, 6, 3, 8, 8, 8], 0, 15) =
= 3
```