On considère des tableaux de nombres dont tous les éléments sont présents exactement trois fois et à suivre, sauf un élément qui est présent une unique fois et que l'on appelle « l'intrus ». Voici quelques exemples :

```
tab_a = [3, 3, 3, 9, 9, 9, 1, 1, 1, 7, 2, 2, 2, 4, 4, 4, 8, 8, 8, 5, 5, 5]
#l'intrus est 7

tab_b = [8, 5, 5, 5, 9, 9, 9, 18, 18, 18, 3, 3, 3]
#l'intrus est 8

tab_c = [5, 5, 5, 1, 1, 1, 0, 0, 0, 6, 6, 6, 3, 8, 8, 8]
#l'intrus est 3
```

On remarque qu'avec de tels tableaux :

- pour les indices multiples de 3 situés strictement avant l'intrus, l'élément correspondant et son voisin de droite sont égaux,
- pour les indices multiples de 3 situés après l'intrus, l'élément correspondant et son voisin de droite - s'il existe - sont différents.

Ce que l'on peut observer ci-dessous en observant les valeurs des paires de voisins marquées par des caractères ^ :

```
[3, 3, 3, 9, 9, 9, 1, 1, 1, 7, 2, 2, 2, 4, 4, 4, 8, 8, 8, 5, 5, 5]
 ^  ^     ^  ^     ^  ^     ^  ^     ^  ^     ^  ^     ^  ^     ^
 0        3        6        9        12       15       18       21
```

Dans des listes comme celles ci-dessus, un algorithme récursif pour trouver l'intrus consiste alors à choisir un indice `i` multiple de 3 situé approximativement au milieu des indices parmi lesquels se trouve l'intrus.

Puis, en fonction des valeurs de l'élément d'indice `i` et de son voisin de droite, à appliquer récursivement l'algorithme à la moitié droite ou à la moitié gauche des indices parmi lesquels se trouve l'intrus.

Compléter la fonction ci-dessous qui met en œuvre cet algorithme.

```
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.
    '''

    if g == d:
        return ...

    else:
        nombre_de_triplets = (d - g)// ...
        indice = g + 3 * (nombre_de_triplets // 2)
        if ... :
            return ...
        else:
            return ...
```

Exemples :

```
>>> 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

>>> trouver_intrus([8, 5, 5, 5, 9, 9, 9, 18, 18, 18, 3, 3, 3], 0, 12)
8

>>> trouver_intrus([5, 5, 5, 1, 1, 1, 0, 0, 0, 6, 6, 6, 3, 8, 8, 8], 0, 15)
3
```