Exercice 1 : On lance indéfiniment une pièce équilibrée. On note X la variable aléatoire égale au rang du lancer où, pour la première fois, on obtient la séquence `Pile-Face-Pile`. On admet que cette séquence apparaît presque sûrement.

Par exemple, si l'on obtient dans cet ordre : Pile, Face, Face, Pile, Pile, Pile, Face, Pile,... alors X prend la valeur 8.

On codera les `Pile` par 1 et les `Face` par 0. On rappelle que si `L` est une liste et k un entier naturel non nul, alors `L[-k]` renvoie le k^{e} élément de la liste `L` en partant de la fin.

Compléter la fonction **Python** ci-dessous afin qu'elle simule les lancers de la pièce jusqu'à l'obtention de la séquence `Pile-Face-Pile`, et qu'elle renvoie le nombre de lancers effectués, c'est-à-dire la valeur que prend X.

```
1  def simulX():
2      X = __2__
3      L = [rd.binomial(1,1/2) for k in range(2)]
4      while True:
5          X = __X + 1__
6          L.append(__rd.binomial(1,1/2)__)
7          if ___L[-1] == 1 and L[-2] == 0 and L[-3] == 1___:
8              return X
```

Exercice 2 : On considère une urne contenant 3 boules rouges et 1 boule verte. On dispose par ailleurs d'une réserve infinie de boules rouges et vertes. On effectue une série infinie de tirages dans cette urne en suivant le protocole suivant :

- si on tire une boule rouge, alors on la remet dans l'urne et on rajoute une nouvelle boule rouge dans l'urne ;
- si on tire une boule verte, alors on la remet dans l'urne et on rajoute deux nouvelles boules vertes dans l'urne.

On note, pour tout entier $n \geq 2$, Y_n la variable aléatoire égale au rang du tirage à l'issue duquel, pour la première fois, le nombre de boules vertes est supérieur ou égal à n. Compléter la fonction **Python** suivante pour qu'elle simule Y_n :

```
1  def simulY(n):
2      r,v = __3,1__
3      Y = 0
4      while __v < n:__
5          Y = Y + 1
6          if rd.randint(1, __r + v + 1__) <= __r:__
7              r = __r + 1__
8          else:
9              v = __v + 2__
10     return Y
```