Exercice 5

1
À l'indice 1 du tableau on trouve 8, à l'indice 3 on trouve 7.
Nous avons 1 < 3 alors que 8 > 7, nous avons donc bien une inversion

2
À l'indice 2 du tableau on trouve 3, à l'indice 3 on trouve 7.
Nous avons 2 < 3 et 3 < 7, nous n'avons donc pas d'inversion

PARTIE A

1a

cas n°1 : 0
cas n°2 : 1
cas n°3 : 2

1b

Si on considère l'élément *b* situé à l'indice *i* dans le tableau tab. La fonction *fonction1* permet de déterminer le nombre d'éléments plus grands que *b* situés dans le tableau à un indice supérieur à *i*.

2

```
nombre_inversions(tab):
      nb_inv = 0
      n = len(tab)
      for i in range(n-1):
            nb_inv = nb_inv + fonction1(tab, i)
      return nb_inv
```

3

L'ordre de grandeur de la complexité en temps de l'algorithme est $O(n^2)$

Partie B
1
Le tri fusion a une complexité en $O(n.\log_2(n))$

2

```
def moitie_gauche(tab):
      n = len(tab)
      nvx_tab = []
      if n==0:
            return []
      mil = n//2
      if n%2 == 0:
            lim = mil
      else :
            lim =mil+1
      for i in range(lim):
            nvx_tab.append(tab[i])
      return nvx_tab
```

une autre possibilité un peu plus concise :

```
def moitie_gauche(tab):
      return [tab[i] for i in range(len(tab)//2+len(tab)%2)]
```

3

```
def nb_inversions_rec(tab):
      if len(tab) > 1:
            tab_g = moitie_gauche(tab)
            tab_d = moitie_droite(tab)
            return nb_inv_tab(tri(tab_g),tri(tab_d)) + nb_inversions_rec(tab_g) \
                  + nb_inversions_rec(tab_d)
      else:
            return 0
```