```python
def _1_admet_point_fixe(t):
    for k in range(len(t)):
        if t[k] == k:
            return True
    return False

def _2_nombre_points_fixes(t):
    nb = 0
    for x in range(len(t)):
        if t[x] == x:
            nb += 1
    return nb

def nombre_points_fixes_bis(t):
    return len([x for x in range(n) if t[x] == x])

def _3_itere(t, x, k):
    a = x
    for j in range(k):
        a = t[a]
    return a

def _4_nb_points_fixes_iteres(t,k):
    nb = 0
    for x in range(len(t)):
        if itere(t, x, k) == x:
            nb += 1
    return nb

# pas du tout efficace :
def _5_admet_attracteur_principal(t):
    attracteur = itere(t, 0, n)
    for x in range(1, n):
        if _3_itere(t, x, n) != attracteur:
            return False
    return True

def _6_temps_de_convergence(t, x):
    if t[x] == x:
        return 0
    else:
        return 1 + _6_temps_de_convergence(t, t[x])

def temps_de_convergence_non_rec(t, x):
    temps = 0
    while t[x] != x:
        x = t[x]
        temps += 1
    return temps

def _7_temps_de_convergence_max(t):
    n = len(t)
    resultat = n*[-1]
    for x in range(n):
        if t[x] == x:
```

```
            resultat[x] = 0
    for x in range(n):
        temps, y = 0, x
        while resultat[y] == -1:
            y = t[y]
            temps += 1
        connu = resultat[y]
        total = temps + connu
        y = x
        while total > connu:
            resultat[y] = total
            total -= 1
            y = t[y]
    return max(resultat)

def _8_est_croissante(t):
    for k in range(len(t) - 1):
        if t[k+1] < t[k]:
            return False
    return True

# invariants : g <= d, t[g] >= g, t[d] <= d
def _9_point_fixe(t):
    g, d = 0, len(t) - 1
    while g < d:
        milieu = (g + d)//2
        valeur = t[milieu]
        if valeur == milieu:
            return milieu
        if valeur < milieu:
            d = milieu - 1
        else:
            g = milieu + 1
    return g

# version récursive
def point_fixe(t):
    n = len(t)
    return point_fixe_aux(t, 0, n)

def point_fixe_aux(t, debut, fin):
    if fin - debut <= 1:
        return debut
    else:
        m = (debut + fin)//2
        if t[m] == m:
            return m
        if t[m] < m:
            return point_fixe_aux(t, debut, m)
        else:
            return point_fixe_aux(t, m+1, fin)
```