Corrigé : points fixes de fonctions à domaine fini (d'après X mp 2013)

Partie I. Recherche de point fixe, cas général

Question 1.

```
def admet_point_fixe(t):
    for x, fx in enumerate(t):
        if fx == x:
            return True
    return False
```

Rappelons que la fonction `enumerate` énumère les couples (indice, élément) d'un tableau.

Question 2.

```
def nb_points_fixes(t):
    s = 0
    for x, fx in enumerate(t):
        if fx == x:
            s += 1
    return s
```

Question 3.

```
def itere(t, x, k):
    y = x
    for i in range(k):
        y = t[y]
    return y
```

Question 4.

```
def nb_points_fixes_iteres(t, k):
    s = 0
    for x in range(len(t)):
        if itere(t, x, k) == x:
            s += 1
    return s
```

Question 5. Considérons une fonction f admettant un attracteur principal z. Sachant que $\operatorname{card} \mathrm{E}_n = n$, pour tout $x \in \mathrm{E}_n$ il existe $i < j$ dans $[\![0, n]\!]$ tel que $f^i(x) = f^j(x)$. La suite $(f^k(x))_{k \geqslant i}$ est alors périodique de période $j - i$. Mais f admet un attracteur principal z donc nécessairement $f^i(x) = z$.
Ceci prouve que si f admet un attracteur principal z alors pour tout $x \in \mathrm{E}_n$, $f^{n-1}(x) = z$. Réciproquement, cette condition implique clairement que z est un attracteur principal, ce qui nous permet de choisir ce critère dans la fonction qui suit.

```
def admet_attracteur_principal(t):
    n = len(t)
    z = itere(t, 0, n-1)
    for x in range(1, n):
        if itere(t, x, n-1) != z:
            return False
    return True
```

Question 6.

```python
def temps_de_convergence(t, x):
    k = 0
    while t[x] != x:
        x = t[x]
        k += 1
    return k
```

Partie II. Recherche efficace de points fixes

Question 7. On observe qu'une fonction est croissante si et seulement si pour tout $x \in [\![0, n-2]\!]$, $f(x) \leqslant f(x+1)$.

```python
def est_croissante(t):
    for x in range(len(t)-1):
        if t[x] > t[x+1]:
            return False
    return True
```

Question 8. Un coût logarithmique suggère un algorithme de recherche dichotomique : on a $0 \leqslant f(0)$ et $f(n-1) \leqslant n-1$. On considère $k = \lfloor n/2 \rfloor$.

- Si $f(k) = k$, la recherche est terminée ;
- Si $f(k) < k$, la recherche se poursuit dans l'intervalle $[\![0, k-1]\!]$;
- Si $k < f(k)$, la recherche se poursuit dans l'intervalle $[\![k+1, n-1]\!]$.

```python
def point_fixe(t):
    i, j = 0, len(t)
    while i + 1 < j:
        k = (i + j) // 2
        if t[k] == k:
            return k
        elif t[k] < k:
            j = k
        else:
            i = k + 1
    return i
```