```python
from random import randint
# Q1
def estPermutation(t):
    n = len(t)
    deja_vu = n*[0]
    for indice in range(n):
        image = t[indice]
        if image < 0 or image >= n or deja_vu[image]
== 1:
            return False
        else:
            deja_vu[image] = 1
    return True

# Q2
def composer(t,u):
    return [t[j] for j in u]

# Q3
def inverser(t):
    n = len(t)
    u = n*[0]
    for k in range(n):
        u[t[k]] = k
    return u

# Q5
def ordre(t):
    n = len(t)
    neutre = [k for k in range(n)]
    u, ord = t, 1
    while u != neutre:
        u = composer(u, t)
        ord += 1
    return ord

# Q6
def periode(t, i):
    p = 1
    j = t[i]
    while j != i:
        j = t[j]
        p += 1
```

```
    return p

# Q7
def estDansOrbite(t, i, j):
    k = t[i]
    while k != i and k != j:
        k = t[k]
    return k == j

# Q8
def estTransposition(t):
    card_s = len([k for k in range(len(t)) if t[k] !=
k])
    return card_s == 2

# Q9
def estCycle(t):
    support = [k for k in range(len(t)) if t[k] != k]
    return support != [] and len(support) ==
periode(t, support[0])

# Q10 avec periode Q6
def periodes(t):
    n = len(t)
    per = n*[0]
    for i in range(n):
        if per[i] == 0:
            p_i = periode(t,i)
            u = i
            for _ in range(p_i): # recopie la valeur
                per[u] = p_i     # de p_i le long de
                u = t[u]         # l'orbite de i
    return per

# Q10 sans periode Q6
def periodes_bis(t):
    n = len(t)
    per = n*[0]
    for i in range(n):
        if per[i] == 0:
            p_i = 1
            j = t[i]
            while j != i:        # détermine la
```

```
                p_i += 1          # période p_i de i
                j = t[j]
            u = i
            for q in range(p_i): # recopie la valeur
                per[u] = p_i     # de p_i le long de
                u = t[u]         # l'orbite de i
    return per

def iterer(t, k):
    u = t
    for q in range(k - 1):
        u = composer(u, t)
    return u

# Q11
def itererEfficace(t,k):
    n = len(t)
    tpk = n*[-1]
    for i in range(n):
        if tpk[i] == -1:
            p_i = periode(t, i)
            k_i = k % p_i
            j = i
            for _ in range(k_i):   # calcul de
                j = t[j]           # de tpk[i]
            u, v = i, j
            for _ in range(p_i):
                tpk[u] = v
                u, v = t[u], t[v]
    return tpk

# Q13
def pgcd(a, b):
    if b == 0:
        return a
    else:
        return pgcd(b, a % b)

# Q14
def ppcm(a, b):
    return (a * b) // pgcd(a, b)

# Q15
```

```
def ordreEfficace(t):
    n = len(t)
    per = periodes(t)
    ordre = 1
    deja_vus = n*[0]
    for i in range(n):
        if not(deja_vus[i]):
            p = per[i]
            ordre = ppcm(ordre, p)
            j = i
            for _ in range(p):
                deja_vus[j] = 1
                j = t[j]
    return ordre




L1 = [0,5,2,3,1,4]
L2 = [1,0,3,2]

# création d'une permutation aléatoire
def permAleatoire(n):
    L = [k for k in range(n)]
    for k in range(n-1):
        m = randint(k,n-1)
        L[k],L[m] = L[m],L[k]
    return L

t = permAleatoire(200)
# print('t : ', t)
print('estPermutation(t) : ', estPermutation(t))
print('estTransposition(t) : ', estTransposition(t))
print('estCycle(t) : ', estCycle(t))
# print('periodes(t) : ', periodes(t))
print('ordreEfficace(t) : ', ordreEfficace(t))
print('ordre(t) : ', ordre(t))
```