Informatique commune - DM1 - corrigé

```
#Question 1
'''dans une liste de longueur n, chaque valeur entre 0 et n-1 doit
apparaitre une unique fois'''
def estPermutation(t):
  n = len(t)
    l = [False]*n #l[j] indique si j a ete rencontre dans t
    for k in range(n):
        if t[k]<0 or t[k]>=n or l[t[k]]:
            return False
        else:
            l[t[k]] = True
    return True

#Question 2
def composer(t,u):
    n = len(t)
    l = [0]*n
    for k in range(n):
        l[k] = t[u[k]]
    return l
'''ou bien avec une definition par comprehension :'''
def composer(t,u):
  return [ t[u[k]] for k in range(len(u)) ]

#Question 3
def inverser(t):
    n = len(t)
    l = [0]*n
    for k in range(n):
        l[t[k]] = k
    return l

#Question 4
'''Seule l'identite est d'ordre 1.
La permutation representee par [1,2,...,n-1,0] est d'ordre n.'''

#Question 5
 def ordre(t):
    n = len(t)
    id = [k for k in range(n)]
    #ou simplement range(n) en python 2
    #ou list(range(n)) en python 3
    k = 1
    tk = t[:]
    while l != i:
        k = k+1
        tk = composer(t,tk)
    return k
```

```python
#Question 6
def periode(t,i):
    k = 1
    tk_i = t[i]
    while tk_i != i:
        tk_i = t[tk_i]
        k += 1
    return k

#Question 7
def EstDansOrbite(t,i,j):
    tk_i = t[i]
    while tk_i != i and tk_i != j:
        tk_i = t[tk_i]
    return tk_i == j

#Question 8
def estTransposition(t):
    n = len(t)
    s = 0
    for k in range(n):
        s += int(t[k] != k)
        #int(True) renvoie 1
        #int(False) renvoie 0
    return s==2

#Question 9
def estCycle(t):
    n = len(t)
    nb_points_non_fixes = 0
    point_non_fixe_trouve = False
    indice_premier_point_non_fixe = 0
    for k in range(n):
        if t[k] != k:
            nb_points_non_fixes += 1
            if not b:
                indice_premier_point_non_fixe = k
                b = True
    return (b and s == periode(t,indice_premier_point_non_fixe) )
```

```
#Question 10
def periodes(t):
    n = len(t)
    orbite_trouvee = [False] * n
    periode = [0] * n
    for k in range(n):
        if not orbite_trouvee[k]:
            orbite_de_k = [k]
            j = t[k]
            while j != k:
              orbite_de_k.append(j)
              j = t[j]
            m = len(orbite_de_k)
            for j in orbite_de_k:
                orbite_trouvee[j] = True
                periode[j] = m
    return periode
'''Le cout est lineaire car les boucles while et for incluses
dans la boucle for principale ne font intervenir chaque valeur
j entre 0 et n-1 qu'une seule fois.'''

#Question 11
def itererEfficace(t,k):
    n = len(t)
    p = periodes(t)
    tk = [0] * n
    for j in range(n):
        r = k % p[j]
        ti_j = j
        for i in range(r):
            ti_j = t[ti_j]
        tk[j] = ti_j
    return tk

#Question 12
'''l'ordre de la permutation suivante vaut 6 (ppcm des longueurs des cycles)
et sa taille vaut 5 :
[1,2,0,4,3]
'''

#Question 13
def pgcd(a,b):
  r = a % b
  u = a
  v = b
  while r != 0:
    u = v
    v = r
    r = u % v
  return v
```

```
#Question 14
def ppcm(a,b):
 return ( a * b ) // pgcd (a , b )

#Question 15
def ordreEfficace ( t ):
  ordre = 1
  n = len ( t )
  for x in range ( n )
    periode_x = periode (t , x )
    if ordre % periode_x != 0:
      ordre = ppcm ( ordre , periode_x )
  return ordre
```