

```
## Piles et files

def creerPile():
    return []

def estVide(p):
    return p == []

def empiler(p, x):
    p.append(x)

def depiler(p):
    if not estVide(p):
        return p.pop()

from collections import deque

def creerFile():
    return deque()

def FileVide(f):
    return f == deque([])

def enfiler(f, x):
    f.append(x)

def defiler(f):
    return f.popleft()

## Exercice 1

def bas1(p):
    """ entrée : p pile
        sortie : renvoie le bas de p sans changer p """
    while not estVide(p):
        a = depiler(p)
        empiler(p, a)
    return a

def bas2(p):
    """ entrée : p pile
        sortie : renvoie le bas de p sans changer p """
    q = creerPile()
    while not estVide(p):
        a = depiler(p)
        empiler(q, a)
    while not estVide(q):
        empiler(p, depiler(q))
    return a

def inside(p, x):
    """ entrée : p pile, x objet
        sortie : booléen indiquant si x apparaît dans p """
    test = False
    q = creerPile()
    while not estVide(p) and test == False:
        a = depiler(p)
        empiler(q, a)
        if a == x:
            test = True
    while not estVide(q):
        empiler(p, depiler(q))
    return test
```

```

def renverse(p):
    """ entrée : p pile
        renverse la pile p
        sortie : None """
    q = creerPile()
    r = creerPile()
    while not estVide(p):
        empiler(q, depiler(p))
    while not estVide(q):
        empiler(r, depiler(q))
    while not estVide(r):
        empiler(p, depiler(r))

def copie(p):
    """ entrée : p pile
        sortie : renvoie une copie de p et ne modifie pas p """
    q = creerPile()
    r = creerPile()
    while not estVide(p):
        empiler(q, depiler(p))
    while not estVide(q):
        a = depiler(q)
        empiler(p, a)
        empiler(r, a)
    return r

## Exercice 2

# Question 1

def scm(s):
    L, i, n = [], 0, len(s)
    while i < n:
        j = i + 1
        while j < n and s[j] >= s[j-1]:
            j += 1
        L.append((i, j - 1))
        i = j
    return L

# Question 2

def fusionner(s, r1, r2):
    d1, f1 = r1
    d2, f2 = r2
    L = []
    while d1 <= f1 and d2 <= f2:
        if s[d1] <= s[d2]:
            L.append(s[d1])
            d1 += 1
        else:
            L.append(s[d2])
            d2 += 1
    if d1 > f1:
        for x in s[d2: f2+1]:
            L.append(x)
    else:
        for x in s[d1: f1+1]:
            L.append(x)
    for i in range(r2[1] - r1[0] + 1):
        s[r1[0] + i] = L[i]

```

```
# Question 3
```

```
def depileFusionneRemplace(s, pile):  
    r2 = depiler(pile)  
    r1 = depiler(pile)  
    fusionner(s, r1, r2)  
    empiler(pile, (r1[0], r2[1]))
```

```
# Question 4
```

```
def alphaTri(s):  
    def longueur(segment):  
        return segment[1] - segment[0] + 1  
    liste_scm = scm(s)  
    nb_scm = len(liste_scm)  
    pile = creerPile()  
    empiler(pile, liste_scm[0])  
    h, i = 1, 1  
    # h est la hauteur de la pile, i l'indice de la dernière scm traitée  
    while i < nb_scm: # première phase  
        if h >= 2:  
            top = depiler(pile)  
            sous_top = depiler(pile)  
            empiler(pile, sous_top)  
            empiler(pile, top)  
            if longueur(sous_top) < 2*longueur(top):  
                depileFusionneRemplace(s, pile)  
                h -= 1  
            else:  
                empiler(pile, liste_scm[i])  
                i += 1  
                h += 1  
        else:  
            empiler(pile, liste_scm[i])  
            i += 1  
            h += 1  
    while h > 1: # deuxième phase  
        depileFusionneRemplace(s, pile)  
        h -= 1
```

```
## Exercice 3
```

```
def longueurfile(f):  
    """ entrée : file f  
        sortie : renvoie la longueur de f """  
    if FileVide(f):  
        return 0  
    g = creerFile()  
    l = 0  
    while not FileVide(f):  
        enfiler(g, defiler(f))  
        l += 1  
    while not FileVide(g):  
        enfiler(f, defiler(g))  
    return l
```

```
def copiefile(f):  
    g = creerFile()  
    h = creerFile()  
    while not FileVide(f):  
        a = defiler(f)  
        enfiler(g, a)  
        enfiler(h, a)  
    while not FileVide(g):  
        enfiler(f, defiler(g))  
    return h
```

Exercice 4

```
def elimination(F, k):
    c = 0
    for _ in range(k-1):
        enfiler(F, defiler(F))
    defiler(F)

def derniers(f, k):
    nb = longueurfile(f)
    while nb > k-1:
        elimination(f, k)
        nb -= 1
    return f

test = creerFile()
test = deque(["Alice", "Bob", "Charles", "Dorine", "Ernest", "Flavien", "Gérard", "Hermione", "Ines",
             , "Jessica"])
print(derniers(test,2))
test = deque(["Alice", "Bob", "Charles", "Dorine", "Ernest", "Flavien", "Gérard", "Hermione", "Ines",
             , "Jessica"])
print(derniers(test,3))
test = deque(["Alice", "Bob", "Charles", "Dorine", "Ernest", "Flavien", "Gérard", "Hermione", "Ines",
             , "Jessica"])
print(derniers(test,4))

for n in range(2,21):
    f = File()
    for k in range(1,n+1):
        f.enfiler(k)
    g = derniers(f, 2).defiler()
    print(n, g)

# Question 2b
""" S'il y a 2n personnes au début, celui qui est en position k au deuxième tour,
    était en position  $k+(k-1) = 2k-1$  au départ.
    S'il y a 2n+1 personnes au début, alors on commence par éliminer les 2,4,6,...,2n et 1
    puis celui désormais en position k était au départ en position 2k+1.
"""

def J(n):
    if n == 1 or n == 2:
        return 1
    if n%2 == 0:
        return 2*J(n//2)-1
    else:
        return 2*J(n//2)+1
```