1 France J2 2021

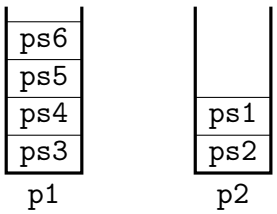
Exercice 1 — Piles et files

1. La structure de données la plus appropriée pour mettre en oeuvre le mode FIFO (First In First Out) est la file (réponse (d)).

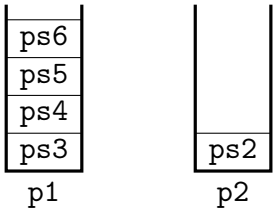
2. Le code suivant convient :

```
def ajouter(lst, proc):
    lst.append(proc)
```

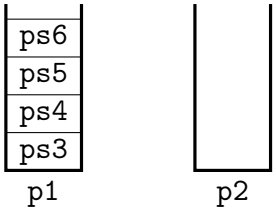
3. `enfiler(file,ps6)` produit :



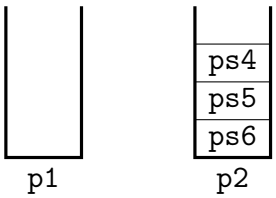
Le premier `defiler(file)` dépile `ps1` de `p2` :



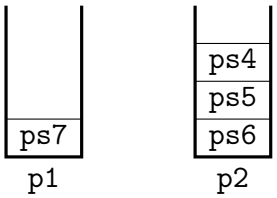
Le deuxième `defiler(file)` dépile `ps2` de `p2` :



Le dernier `defiler(file)` dépile tout `p1` dans `p2` avant de dépiler `ps3` de `p2` :



Finalement le contenu final après `enfiler(file,ps7)` des deux piles est :



4. (a) La file est vide si les deux piles représentant la file sont vides toutes les deux, ainsi :

```
def est_vide(f):
    return pile_vide(f[0]) and pile_vide(f[1])
```

(b) Quoiqu'il arrive, une nouvel élément est ajouté au sommet de la pile p1.

```
def enfiler(f, elt):
    empiler(f[0], elt)
```

(c) Deux cas sont à gérer selon si la pile p2 est vide ou non.

```
def defiler(f):
    assert not est_vide(f), "la file est vide, aucun élément à défiler"
    if pile_vide(f[1]):
        while not pile_vide(f[0]):
        empiler(f[1],depiler(f[0]))
    return depiler(f[1])
```

2 Amérique du nord 2021

Exercice 2 — Pile, file et programmation

1. (a) La file F sera vide et la pile P sera :

"rouge"
"vert"
"jaune"
"rouge"
"jaune"

(b) La fonction suivante convient :

```
def taille_file(F):
    t = 0
    ft = creer_file_vide()
    while not est_vide(F):
        t = t + 1
        enfiler(ft, defiler(F))
    while not est_vide(ft):
        enfiler(F, defiler(ft))
    return t
```

2. La fonction suivante convient :

```
def former_pile(F):
    p = creer_pile_vide()
    pt = creer_pile_vide()
    while not est_vide(F):
```

```
        empiler(pt,defiler(F))
    while not est_vide(pt):
        empiler(p,depiler(pt))
    return p
```

3. La fonction suivante convient :

```
def nb_elements(F, ele):
    nb = 0
    ft = creer_file_vide()
    while not est_vide(F):
        x = defiler(F)
        if x == ele:
            nb = nb + 1
        enfiler(ft, x)
    while not est_vide(ft):
        enfiler(F, defiler(ft))
    return nb
```

4. La fonction suivante convient :

```
def verifier_contenu(F, nb_rouge, nb_vert, nb_jaune):
    return nb_elements(F, "rouge") <= nb_rouge and nb_elements(F, "vert") <= nb_vert
            and nb_elements(F, "jaune") <= nb_jaune
```

3 Amérique du Nord J1 2022

Exercice 3 — Files et tableaux

1. (a) Les éléments 15, 17 et 14 ont été enfilés dans cette ordre, donc c'est la Proposition 2 qui est correcte.

   (b) Les instructions suivantes conviennent :

```
f = creer_file_vide()
enfiler(f, 15)
enfiler(f, 17)
enfiler(f, 14)
```

2. La fonction suivante convient :

```
def longueur_file(F):
    G = creer_file_vide()
    n = 0
    while not(est_vide(F)):
        v = defiler(F)
        n = n + 1
```

```
        enfiler(G, v)
    while not(est_vide(G)):
        v = defiler(G)
        enfiler(F, v)
    return n
```

3. La fonction suivante convient :

```
def variations(F):
    taille = longueur_file(F)
    if taille == 1 :
        return []
    else:
        tab = [0 for k in range(taille - 1)]
        element1 = defiler(F)
        for i in range(taille - 1):
            element2 = defiler(F)
            tab[i]=element2 - element1
            element1 = element2
    return tab
```

4. La fonction suivante convient :

```
def nombre_baisses(tab):
    mini = tab[0]
    nbr = 0
    for v in tab:
        if v < 0:
            nbr = nbr + 1
        if v < mini:
            mini = v
    if nbr == 0:
        return (0,0)
    else:
        return (nbr, mini)
```

4 Centre étrangers J1 2022

EXERCICE 4 — Files et POO

1. Le code suivant convient :

```
panier1.enfiler((31002, "café noir", 1.50, 50525))
```

2. Le code suivant convient :

```
def remplir(self, panier_temp):
    while not panier_temp.est_vide() :
        article = panier_temp.defiler()
        self.enfiler(article)
```

3. Le code suivant convient :

```
def prix_total(self):
    p_temp = Panier()
    montant = 0
    while not self.est_vide() :
        article = self.defiler()
        montant = montant + article[2]
        p_temp.enfiler(article)
    while not p_temp.est_vide() :
        article = p_temp.defiler()
        self.enfiler(article)
    return montant
```

4. Le code suivant convient :

```
def horaire_scan(self):
    if self.est_vide():
        return 0
    premier_article = self.defiler()[3]
    dernier_article = premier_article
    while not self.est_vide() :
        dernier_article = self.defiler()[3]
    return dernier_article - premier_article
```

5 Mayotte J1 2022

Exercice 5 — Piles et files

1. (a) On a la file f suivante :

1	4	3	8	2

(b) On a la pile p suivante :

5
8
6
2

(c) On a la pile p et la file f suivantes :

8	5	4	3	8	2	1

6
2

(d) On a la pile `p` et la file `f` suivantes :

4	3	8

2
1
5
8
6
2

2. A chaque tour de la première boucle, la file `f` perd un élément jusqu'à ce qu'elle soit vide. Puis, à chaque tour de la seconde boucle, la file `f` gagne un élément. A l'issue de la fonction, le contenu de la file `f` est | 4 | 3 | 2 | 1 |.
La pile renvoyée par la fonction `mystere` est vide.

3. (a) On a le tableau suivant :

f	2 1 3	2 1	3 2	3	2 3	1 2 3
p	⌴	3	1	2 1	1	⌴

(b) Cet algorithme permet de trier dans l'ordre décroissant (plus grand élément en tête de file) une file composée de trois éléments (pour quatre éléments ou plus cela ne fonctionne pas !?). Si on prend une file contentant au départ plus de trois éléments, on peut dire que cet algorithme permet de mélanger cette file.

6 France Sept 2022

Exercice 6 — Files et programmation générale

1. Il s'agit du principe FIFO et donc de la situation 2.

2. (a)
   - ⋆ `v` est une file `[Client3, Client2, Client1]` (`Client1` correspond à la tête de file) ;
   - ⋆ `F` est une file `[Client4]` ;
   - ⋆ `val` contient la valeur `Prioritaire`

   (b) Le code suivant convient :

```
def longueur_file(F) :
    V = creer_file_vide()
    n = 0
    while not est_vide (F):
        n = n + 1
        val = defiler(F)
        enfiler(V, val)
    while not est_vide(V):
        val = defiler(V)
        enfiler(F, val)
    return n
```

(c) Le code suivant convient :

```
def compter_prio(F):
    V = creer_file_vide()
    n = 0
    while not est_vide (F):
        val = defiler(F)
        enfiler(V,val)
        if val == 'Prioritaire':
            n = n + 1
    while not est_vide(V):
        val = defiler(V)
        enfiler(F, val)
    return n
```

7 Centres-étrangers J1 2023

Exercice 7 — Files

Q1.
```
def ajout(f):
    couleurs = ("bleu", "rouge", "jaune", "vert")
    indice = randint(0, 3)
    enfiler(f, couleurs[indice])
    return f
```

Q2.
```
def vider(f):
    while not est_vide(f):
        defiler(f)
```

Q3.

```
def affich_seq(sequence):
    stock = creer_file_vide()
    ajout(sequence)
    while not est_vide(sequence):
        c = defiler(sequence)
        affichage(c)
        time.sleep(0.5)
        enfiler(stock, c)
    while not est_vide(stock):
        enfiler(sequence, defiler(stock))
```

Q4. **(a)**

```
def tour_de_jeu(sequence):
    affich_seq(sequence) #ZONE A
    stock = creer_file_vide()
    while not est_vide(sequence):
        c_joueur = saisie_joueur()
        c_seq = defiler(sequence) #ZONE B
        if c_joueur == c_seq:
            enfiler(stock, c_seq) #ZONE C
        else :
            vider(sequence) #ZONE D
    while not est_vide(stock): #ZONE E
        enfiler(sequence, defiler(stock)) #ZONE F
```

(b)

```
def tour_de_jeu_modif(sequence):
    affich_seq(sequence)
    t_ini = taille(sequence)
    stock = creer_file_vide()
    while not est_vide(sequence):
        c_joueur = saisie_joueur()
        c_seq = defiler(sequence)
        if c_joueur == c_seq:
            enfiler(stock, c_seq)
        else: vider(sequence)
    while not est_vide(stock):
        enfiler(sequence, defiler(stock))
    if t_ini != taille(sequence) :
        vider(sequence)
    tour_de_jeu_modif(sequence)
```

.