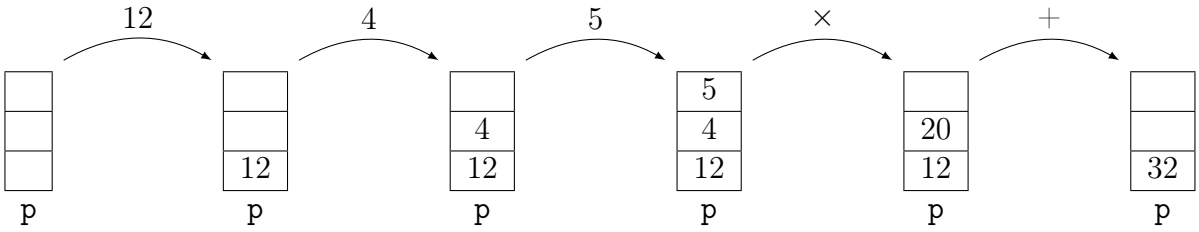
LYCÉE MARIE CURIE **0NSI**
2022–2023

Devoir surveillé n°3 – NSI
Correction

Exercice 1

1. Le schéma est le suivant :



2. (a) la variable `temp` contient la valeur 25.
   (b) La pile `p1` peut être représentée ainsi :

25
3
7

3. On a :

```
def addition(p):
    v1 = depiler(p)
    v2 = depiler(p)
    empiler(p, v1+v2)
```

4. Les lignes de codes sont les suivantes :

```
p=pile_vide()
empiler(p,3)
empiler(p,5)
addition(p)
empiler(p,7)
multiplication(p)
```

Exercice 2

1. (a) La file `f` est :

1	4	3	8	2

   (b) La pile `p` est :

5
8
6
2

   (c) La file `f` et la pile `p` sont :

8	5	4	3	8	2	1

et

6
2

   (d) La file `f` et la pile `p` sont :

4	3	8

et

2
1
5
8
6
2

2. la variable `f` contient : | 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	2	3,2	3,2	3	3	2,3	1,2,3
`p`	⌊ ⌋	⌊3⌋	⌊3⌋	⌊ ⌋	⌊1⌋	⌊1⌋	⌊2 1⌋	⌊1⌋	⌊ ⌋

   (b) Cette fonction est assez complexe, et ne fait pas forcément toujours tout à fait ce que l'on voudrait.

   L'algorithme parcourt la file en mettant la première valeur dans une pile.

   Si la valeur de la file considérée est plus petite que le sommet de la pile, on dépile (en enfilant) jusqu'à ce que la pile soit vide ou que la valeur devienne plus grande que le sommet de la pile, puis on empile la valeur.

   Quand on a parcouru tous les éléments de la file, on enfile tous les éléments restants de la pile.

   On peut remarquer que la pile est toujours triée par ordre croissant (du bas vers le sommet). Quand on enfile ses valeurs, on a donc des valeurs triées dans la file.

   Il y a ainsi un tri (toujours croissant) par morceaux.

   On peut remarquer que si on exécute la fonction sur une liste triée, l'ordre de la liste ne change pas (on ne fait qu'enfiler la valeur que l'on vient d'empiler).

   Si on exécute au contraire la fonction sur une liste triée à l'envers, elle la trie dans l'autre sens, puisqu'on empile à chaque fois directement la valeur, puis à la fin on va tout empiler.

   L'algorithme permet de trier certaines listes, comme on peut le voir sur cette page. Mais l'algorithme ne permet pas de trier toutes les listes. En regardant le code JavaScript, on voit que la liste n'est pas vraiment triée au hasard de manière quelconque. En particulier, on ne tombera par exemple jamais sur une liste terminant par 5,6. Cela obligerait à enfiler 6 avant d'avoir pu enfiler 8 puis 7, rendant impossible la résolution du problème. En fait, la manière de mélanger la liste consiste à faire des opérations contraires à celles effectuées par l'algorithme, autrement dit la résolution de l'exercice consiste à faire les opérations dans le sens contraire de celles faites pour mélanger.

Voici du code permettant de tester la fonction `knuth` de l'exercice 2 :

```
## Exercice 2 3. (fonction knuth)

class Pile:
    '''structure de pile'''
    def __init__(self):
        self.contenu = []
    def est_vide(self):
        return len(self.contenu) == 0
    def empiler(self, e):
        self.contenu.append(e)
    def depiler(self):
        if self.est_vide():
            raise IndexError('dépiler sur une pile vide')
        else:
            return self.contenu.pop()
    def sommet(self):
        return self.contenu[-1]

def creer_pile_vide():
    return Pile()
def est_pile_vide(p):
    return p.est_vide()
def empiler(p, e):
    p.empiler(e)
def depiler(p):
    return p.depiler()
def sommet(p):
    return p.sommet()

class Cellule:
    '''Une cellule d'une liste chaînée'''
    def __init__(self, v, s):
        self.valeur = v
        self.suivante = s
    def __str__(self): # Affichage à l'envers
        if self is None:
            return ""
        else:
            if self.suivante is not None:
                return str(self.suivante)+", "+str(self.valeur)
            else:
                return str(self.valeur)

def longueur(lst):
    '''renvoie la longueur de la liste lst'''
    if lst is None:
        return 0
    else:
        return 1 + longueur(lst.suivante)
```

```python
class File:
    '''structure de file'''
    def __init__(self):
        self.tete = None
        self.queue = None
    def est_vide(self):
        return self.tete is None
    def ajouter(self, e):
        c = Cellule(e, None)
        if self.est_vide():
            self.tete = c
        else:
            self.queue.suivante = c
        self.queue = c
    def retirer(self):
        if self.est_vide():
            raise IndexError('retirer sur une file vide')
        else:
            v = self.tete.valeur
            self.tete = self.tete.suivante
            if self.tete is None:
                self.queue = None
            return v
    def taille(self):
        return longueur(self.tete)
    def __str__(self):
        return str(self.tete)

def creer_file_vide():
    return File()
def est_file_vide(f):
    return f.est_vide()
def enfiler(f, e):
    return f.ajouter(e)
def defiler(f):
    return f.retirer()
def taille_file(f):
    return f.taille()

def knuth(f):
    p=creer_pile_vide()
    N=taille_file(f)
    for i in range(N):
        if est_pile_vide(p):
            empiler(p, defiler(f))
        else:
            e = defiler(f)
            if e >= sommet(p):
                empiler(p, e)
            else:
                while not est_pile_vide(p) and e < sommet(p):
                    enfiler(f, depiler(p))
```

```
                empiler(p, e)
    while not est_pile_vide(p):
        enfiler(f, depiler(p))

print("Cas de l'exercice")
f = creer_file_vide()
enfiler(f,3)
enfiler(f,1)
enfiler(f,2)
print(f)
knuth(f)
print(f)

print("Cas montrant que ça ne trie pas globalement")
f = creer_file_vide()
enfiler(f,2)
enfiler(f,1)
enfiler(f,3)
print(f)
knuth(f)
print(f)

print("Cas de liste croissante")
f = creer_file_vide()
enfiler(f,6)
enfiler(f,5)
enfiler(f,4)
enfiler(f,3)
enfiler(f,2)
enfiler(f,1)
print(f)
knuth(f)
print(f)

print("Cas de liste décroissante")
f = creer_file_vide()
enfiler(f,1)
enfiler(f,2)
enfiler(f,3)
enfiler(f,4)
enfiler(f,5)
enfiler(f,6)
print(f)
knuth(f)
print(f)

print("quatrième cas : liste quelconque ; plusieurs étapes")
f = creer_file_vide()
enfiler(f,3)
enfiler(f,5)
enfiler(f,1)
enfiler(f,6)
```

```
enfiler(f,4)
enfiler(f,2)
print(f)
knuth(f)
print(f)
knuth(f)
print(f)
knuth(f)
print(f)
```