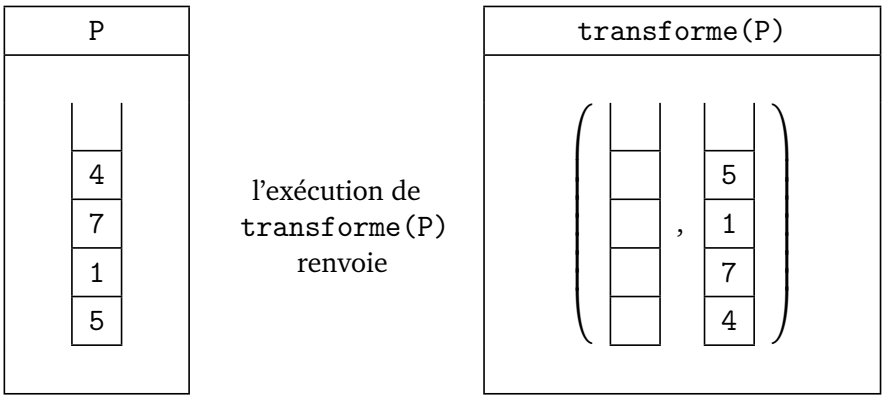
EXERCICE 1 Centres-Etrangers 2021 - Piles - 4 points

1. On a le schéma suivant :

	Etape 0 Pile d'origine P	Etape 1 `empiler(P,8)`	Etape 2 `depiler(P)`	Etape 3 `est_vide(P)`
				
Valeur renvoyée		None	8	False

2. On a le schéma suivante :



3. La fonction suivante convient :

```
def maximum(P):
    m = 0
    Q = creer_pile()
    while not est_vide(P):
        v = depiler(P)
        empiler(q,v)
        if v > m:
            m = v
    while not est_vide(Q):
        e=depiler(Q)
        empiler(P,e)
    return m
```

4. (a) Il suffit de mettre place une boucle qui s'arrêtera quand la pile P sera vide. À chaque tour de boucle, on dépile P, on empile les valeurs précédemment dépilées dans une pile auxiliaire Q et on incrémente un compteur de 1. Une fois la boucle terminée, on crée une nouvelle boucle où on dépile Q et on empile P avec les valeurs dépilées (l'idée est de retrouver l'état originel de P). Il suffit ensuite de renvoyer la valeur du compteur.

   (b) La fonction suivante convient :

```
def taille(P):
    cpt = 0
    Q = creer_pile()
    while not est_vide(P):
        v = depiler(P)
        empiler(Q,v)
        cpt = cpt + 1
    while not est_vide(Q):
        v = depiler(Q)
        empiler(P,v)
    return cpt
```

EXERCICE 2 — France J1 - 2022 - Structure de données - 4 points

Partie A : Expression correctement parenthésée

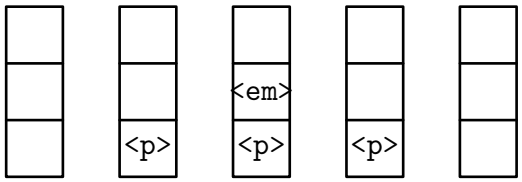
1. Quand on traite les éléments dans l'ordre d'arrivée il s'agit d'une file (FIFO : premier arrivé, premier sorti).

2. Pour l'expression simplifiée B, les valeurs successives prises par la variable `controleur` sont 1, 2, 3, 2, 3, 2.
   Pour l'expression simplifiée C, les valeurs successives prises par la variable `controleur` sont 1, 2, 1, 0, -1, 0.

3. On peut compléter la ligne 13 avec : `if controleur < 0`
   Le test suivant convient pour la ligne 16 : `if controleur == 0`.

Partie B : Texte correctement balisé

4. (a) Voici la représentation de l'état de la pile pour l'expression :

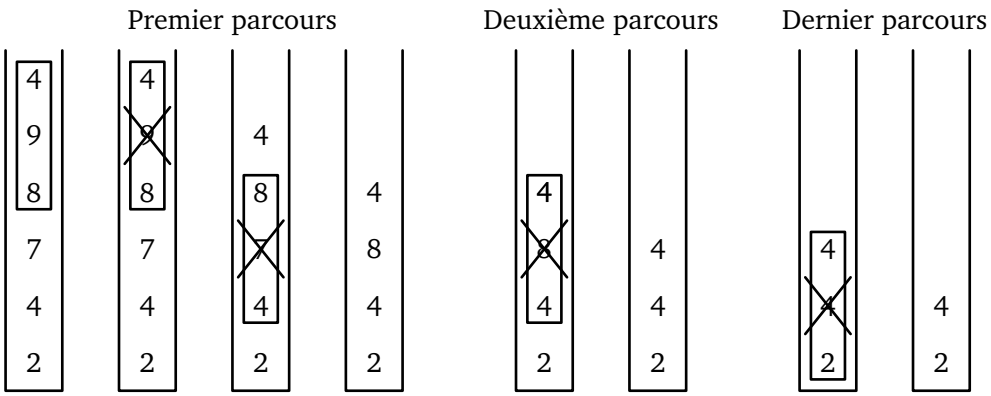


   (b) Si la pile est vide à la fin du traitement alors le texte est correctement balisé.

5. Une expression HTML correctement balisée doit contenir autant de balises ouvrantes que de balises fermantes. Puisqu'on empile uniquement les balises ouvrantes, si une expression HTML correctement 12 balises, dans le pire des cas elles sont imbriquées les unes dans les autres, alors le nombre maximum d'éléments que contiendra la pile est 6.

EXERCICE 3 — France J2 - 2022 - Structure de données - 4 points

1. (a) Voici les différentes étapes de réduction de la pile :



(b) La pile B est la pile gagnante.

2. On complète les lignes 5 à 7 de la façons suivante :

```
if a % 2 != c % 2:
    empiler(p, b)
    empiler(p, a)
```

3. (a) La taille minimale que doit avoir une pile pour être réductible est 3.

(b) On complète la ligne 3 avec : `while taille(p) >= 3:`
et les lignes 8 et 9 par :

```
e = depiler(q)
empiler(p, e)
```

4. Le code suivant convient :

```
def jouer(p):
    q = parcourir_pile_en_réduisant(p)
    if taille(p) == taille(q):
        return p
    else:
        return jouer(q)
```

EXERCICE 4 — Mayotte J2 - 2022 - Piles - 4 points

1. On a le schéma suivant :

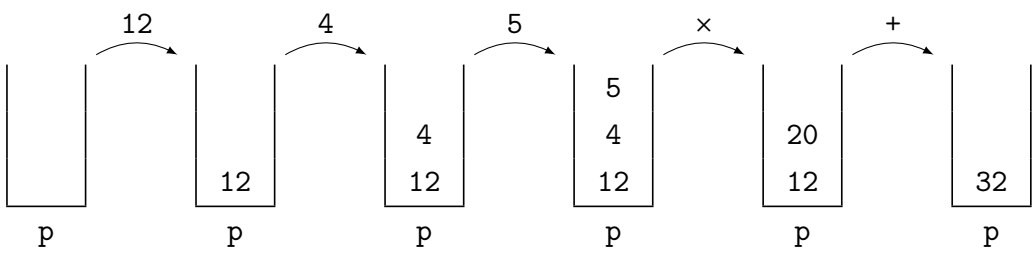
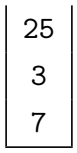


Schéma descriptif des différentes étapes d'exécution

2. (a) La fonction `top` renvoie la valeur au sommet de la pile. Donc, La variable `temp` contient la valeur 25.

   (b) On a la pile suivante :



3. La fonction suivante convient :

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

4. Le code suivant convient :

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

EXERCICE 5 Polynésie J1 - 2022 - Structure de données, piles - 4 points

1. On complète la ligne 6 ainsi : `if e1 > e2:` et la ligne 8 avec : `e1 = e2`

2. (a) La valeur renvoyée par l'appel `A.est_triee()` renvoie `False` car 4 est plus grand que 3.

   (b) Après l'exécution de cette instruction, `A` vaut `[1,2]` (les élements 3 et 4 ont été dépilés pour comparaison).

3. On complète la ligne 9 ainsi : `maxi = elt` et la ligne 11 avec : `q.empiler(elt)`

4. (a) **Initialisation** `B` vaut `[9, -7, 8, 12]` et `q` vaut `[]`

   Itération 1 `B` vaut `[9, -7, 8]` et `q` vaut `[4]`

   Itération 2 `B` vaut `[9, -7]` et `q` vaut `[4, 8]`

   Itération 3 `B` vaut `[9]` et `q` vaut `[4, 8, -7]`

   Itération 4 `B` vaut `[]` et `q` vaut `[4, 8, -7, 9]`

   (b) Avant l'exécution de la ligne 14, la pile `q` est vide, elle vaut `[]` et la pile `B` contient la liste `[9, -7, 8, 4]`.

   (c) Si `B = [12, 5, 10]` alors le résultat obtenu est `[10,5]`, l'ordre des éléments est modifié.

5. (a) **Avant la ligne 3** `B` vaut `[1, 6, 4, 3, 7, 2]` et `q` vaut `[]`

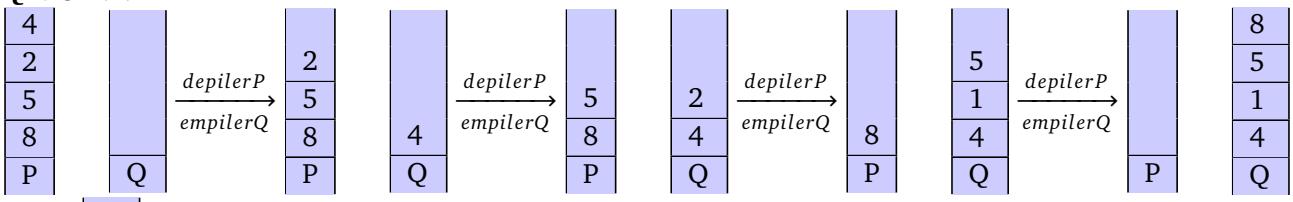
   Avant la ligne 5 `B` vaut `[]` et `q` vaut `[7, 6, 4, 3, 2, 1]`

À la fin B vaut `[1, 2, 3, 4, 5, 6, 7]` et q vaut `[]`

(b) Les données de la liste représentant la pile B sont dans l'ordre croissant donc la pile contient les éléments dans l'ordre décroissant. La fonction `traiter` trie les données.
En effet, la boucle `while` à la ligne 3 permet d'empiler dans la pile q les éléments de la pile B du plus grand au plus petit et la boucle `while` de la ligne 5 permet d'empiler de nouveau tous les éléments de q dans la pile B, ce qui a pour effet d'empiler le plus petit en premier et le plus grand en dernier.

EXERCICE 6 Sujet-0 - Piles - 4 points

Q1. On a :



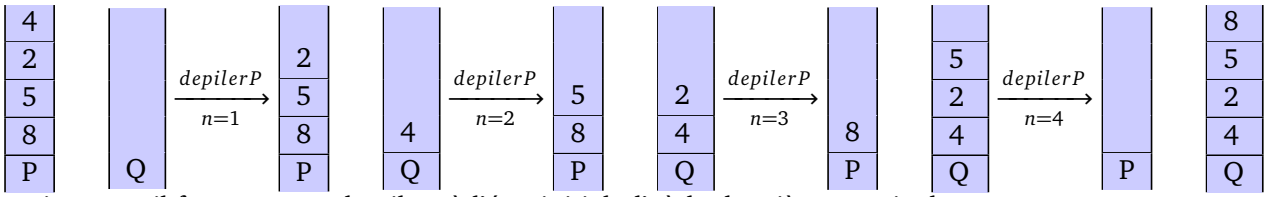
Donc,

Q2. Programme demandé :

```
def hauteur_pile(P):
    Q=creer_pile_vide()
    n=0
    while not est_vide(P)
        n+=1
        x=depiler(P)
        empiler(Q,x)
    while not est_vide(Q):
        x=depiler(Q)
        empiler(P,x)
    return n
```

➔ **Explication :**

```
Q=creer_pile_vide()
n=0
while not est_vide(P):
    n+=1
    x=depiler(P)
    empiler(Q,x)
```



Maintenant il faut remettre la pile P à l'état initial, d'où la deuxième partie du programme :

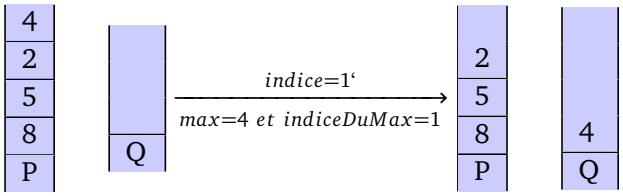
```
while not est_vide(Q):
    x=depiler(Q)
    empiler(P,x)
```

Q3. Programme demandé :

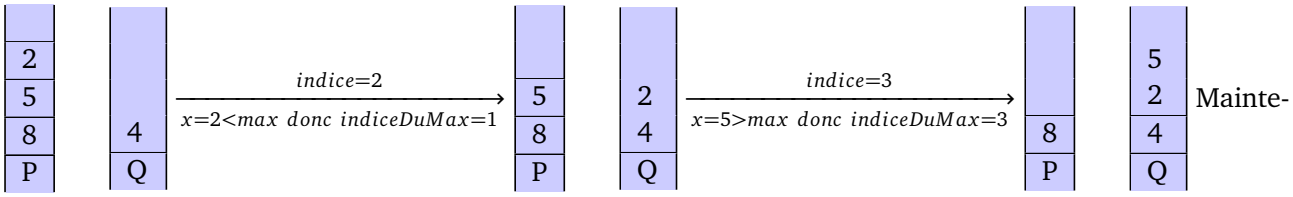
```
def max_pile(P,i):
    Q=creer_pile_vide()
    indice=1
    indiceDuMax=1
    max=depiler(P)
    empiler(Q,max)
    while indice<i:  #On a inégalité stricte car on par de l'indice 2 pour la boucle
        x=depiler(P)
        indice+=1
        if x>max:
           max=x
           indiceDuMax=indice
        empiler(Q,x)
    while not est_vide(Q):
        x=depiler(Q)
        empiler(P,x)
   return indiceDuMax
```

➔ **Explication :**

```
Q=creer_pile_vide()
indice=1
indiceDuMax=1
max=depiler(P)
empiler(Q,max)
```




```
while indice<i:  #On a inégalité stricte car on par de l'indice 2 pour la boucle
    x=depiler(P)
    indice+=1
    if x>max:
        max=x
        indiceDuMax=indice
        empiler(Q,x)
```



Mainte-

nant il faut remettre la pile P à l'état initial, d'où la deuxième partie du programme :

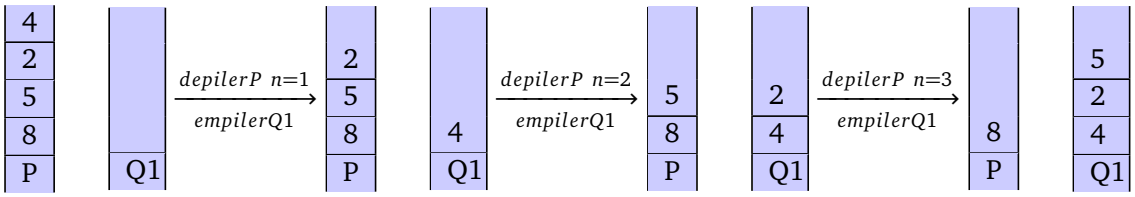
```
while not est_vide(Q):
    x=depiler(Q)
    empiler(P,x)
```

Q4. Programme demandé :

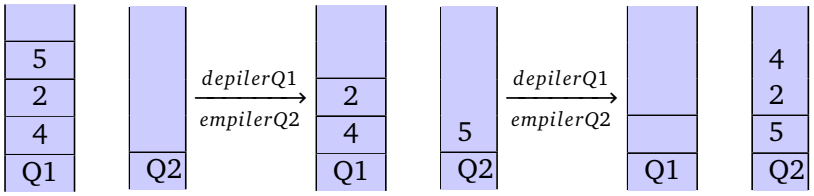
```
def retourner(P,j):
    Q1=creer_pile_vide()
    Q2=creer_pile_vide()
    k=1
    while k<=j:
        x=depiler(P)
        empiler(Q1,x)
        k+=1
    while not est_vide(Q1):
        el=depiler(Q1)
        empiler(Q2,el)
    while not est_vide(Q2):
        y=depiler(Q2)
        empiler(P,y)
    return P
```

➔ **Explication :**

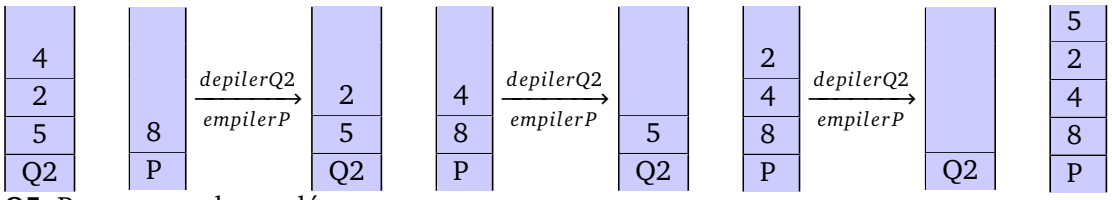
```
while k<=j:
    x=depiler(P)
    empiler(Q1,x)
    k+=1
```




```
while not est_vide(Q1):
    el=depiler(Q1)
    empiler(Q2,el)
```




```
while not est_vide(Q2):
    y=depiler(Q2)
    empiler(P,y)
```



Q5. Programme demandé :

```
def tri_crepe(P):
    k=hauteur_pile(P)
    while k >0:
        i=max_pile(P,k)
        retourner(P,i)
        retourner(P,k)
        k-=1
    return P
```

EXERCICE 7 — France Sept 2021 - POO - 4 points

1. (a) Les deux assertions sont les suivantes :

```
assert arome in ['fraise' , 'abricot', 'vanille', 'aucun'], "Cet arôme est
assert duree > 0 and duree < 366, "la durée doit être comprise entre 1 et
```

(b) Le genre associé à `Mon_Yaourt` sera `aromatise`.

(c) La méthode suivante convient :

```
def GetArome(self):
    return self.__arome
```

2. La fonction suivante convient :

```
def SetArome(self, arome):
    assert arome in ['fraise' , 'abricot', 'vanille', 'aucun']], "Cet arôm
    self.__arome = arome
    self.__SetGenre(arome)
```

3. (a) La fonction suivante convient :

```
def empiler(p, Yaourt):
    p.append(Yaourt)
    return p
```

(b) La fonction suivante convient :

```
def depiler(p):
    return p.pop()
```

(c) La fonction suivante convient :

```
def estVide(p):
 return len(p)==0
```

(d) Le bloc d'instructions affiche :

```
24
False
```