

### Exercice 5 (4 points).

*Cet exercice porte sur la notion de pile, de file et sur la programmation de base en Python.*

Les interfaces des structures de données abstraites `Pile` et `File` sont proposées ci-dessous.

On utilisera uniquement les fonctions ci-dessous :

#### Structure de données abstraite : Pile

Utilise : `Élément`, `Booléen`

##### Opérations :

- `creer_pile_vide` :  $\emptyset \rightarrow \text{Pile}$   
`creer_pile_vide()` renvoie une pile vide
- `est_vide` : `Pile`  $\rightarrow$  `Booléen`  
`est_vide(pile)` renvoie `True` si `pile` est vide, `False` sinon
- `empiler` : `Pile`, `Élément`  $\rightarrow \emptyset$   
`empiler(pile, element)` ajoute `element` à la pile `pile`
- `depiler` : `Pile`  $\rightarrow$  `Élément`  
`depiler(pile)` renvoie l'élément au sommet de la `pile` en le retirant de la `pile`

#### Structure de données abstraite : File

Utilise : `Élément`, `Booléen`

##### Opérations :

- `creer_file_vide` :  $\emptyset \rightarrow \text{File}$   
`creer_file_vide()` renvoie une file vide
- `est_vide` : `File`  $\rightarrow$  `Booléen`  
`est_vide(file)` renvoie `True` si `file` est vide, `False` sinon
- `enfiler` : `File`, `Élément`  $\rightarrow \emptyset$   
`enfiler(file, element)` ajoute `element` dans la file `file`
- `defiler` : `File`  $\rightarrow$  `Élément`  
`defiler(file)` renvoie l'élément au sommet de la file `file` en le retirant de la file `file`

1. (a) On considère la file `F` suivante :

enfilement  $\longrightarrow$  "rouge" "vert" "jaune" "rouge" "jaune"  $\longrightarrow$  défilement

Quel sera le contenu de la pile `P` et de la file `F` après l'exécution du programme Python suivant ?

```
1 P = creer_pile_vide()
2 while not(est_vide(F)):
3     empiler(P, defiler(F))
```

- (b) Créer une fonction *taille\_file* qui prend en paramètre une file *F* et qui renvoie le nombre d'éléments qu'elle contient. Après appel de cette fonction la file *F* doit avoir retrouvé son état d'origine.

```
1 def taille_file(F):  
2     """File -> Int"""
```

2. Écrire une fonction *former\_pile* qui prend en paramètre une file *F* et qui renvoie une pile *P* contenant les mêmes éléments que la file.

Le premier élément sorti de la file devra se trouver au sommet de la pile ; le deuxième élément sorti de la file devra se trouver juste en-dessous du sommet, etc.

**Exemple :** si *F* = "rouge" "vert" "jaune" "rouge" "jaune" alors l'appel *former\_pile(F)* va renvoyer la pile *P* ci-dessous :

*P* = 

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

3. Écrire une fonction *nb\_elements* qui prend en paramètres une file *F* et un élément *elt* et qui renvoie le nombre de fois où *elt* est présent dans la file *F*.

Après appel de cette fonction la file *F* doit avoir retrouvé son état d'origine.

4. Écrire une fonction *verifier\_contenu* qui prend en paramètres une file *F* et trois entiers : *nb\_rouge*, *nb\_vert* et *nb\_jaune*.

Cette fonction renvoie le booléen *True* si "rouge" apparaît au plus *nb\_rouge* fois dans la file *F*, "vert" apparaît au plus *nb\_vert* fois dans la file *F* et "jaune" apparaît au plus *nb\_jaune* fois dans la file *F*. Elle renvoie *False* sinon. On pourra utiliser les fonctions précédentes.