

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 : <ul style="list-style-type: none"> • creer_pile_vide : $\emptyset \rightarrow \text{Pile}$ <code>creer_pile_vide()</code> renvoie une pile vide • est_vide : $\text{Pile} \rightarrow \text{Booléen}$ <code>est_vide(pile)</code> renvoie True si pile est vide, False sinon • empiler : $\text{Pile}, \text{Élément} \rightarrow \emptyset$ <code>empiler(pile, element)</code> ajoute element à la pile pile • depiler : $\text{Pile} \rightarrow \text{Élément}$ <code>depiler(pile)</code> renvoie l'élément au sommet de la pile en le retirant de la pile

Structure de données abstraite : File
Utilise : Élément, Booléen
Opérations : <ul style="list-style-type: none"> • creer_file_vide : $\emptyset \rightarrow \text{File}$ <code>creer_file_vide()</code> renvoie une file vide • est_vide : $\text{File} \rightarrow \text{Booléen}$ <code>est_vide(file)</code> renvoie True si file est vide, False sinon • enfiler : $\text{File}, \text{Élément} \rightarrow \emptyset$ <code>enfiler(file, element)</code> ajoute element dans la file file • defiler : $\text{File} \rightarrow \text{Élément}$ <code>defiler(file)</code> 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.