Devoir Surveillé - 2 h - n°1

Exercice 1 - *Ameique Nord - 2021 - sujet 1 - Pile et File* 14 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 :

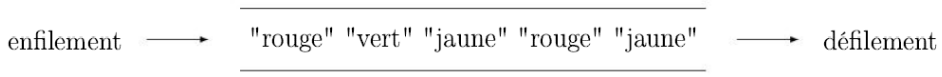
Structure de données abstraite : Pile
Utilise : Élément, Booléen

Opérations :
• `creer_pile_vide` : $\emptyset \longrightarrow$ `Pile`
`creer_pile_vide()` renvoie une pile vide
• `est_vide` : `Pile` $\longrightarrow$ `Booléen`
`est_vide(pile)` renvoie True si `pile` est vide, False sinon
• `empiler` : `Pile, Élément` $\longrightarrow \emptyset$
`empiler(pile, element)` ajoute `element` à la pile `pile`
• `depiler` : `Pile` $\longrightarrow$ `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` : $\emptyset \longrightarrow$ `File`
`creer_file_vide()` renvoie une file vide
• `est_vide` : `File` $\longrightarrow$ `Booléen`
`est_vide(file)` renvoie True si `file` est vide, False sinon
• `enfiler` : `File, Élément` $\longrightarrow \emptyset$
`enfiler(file, element)` ajoute `element` dans la file `file`
• `defiler` : `File` $\longrightarrow$ `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 :



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.

```
def taille_file(F):
    """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 :

$$P = \begin{array}{|c|}\hline \text{"jaune"} \\ \hline \text{"rouge"} \\ \hline \text{"jaune"} \\ \hline \text{"vert"} \\ \hline \text{"rouge"} \\ \hline \end{array}$$

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.

Exercice 2 - *épreuve pratique 2021 - sujet 37* 7 points

Programmer la fonction `verifie` qui prend en paramètre un tableau de valeurs numériques non vide et qui renvoie `True` si ce tableau est trié dans l'ordre croissant, `False` sinon.

Exemples :

```
>>> verifie([0, 5, 8, 8, 9])
True
>>> verifie([8, 12, 4])
False
>>> verifie([-1, 4])
True
>>> verifie([5])
True
```

Exercice 3 - *épreuve pratique 2021 - sujet 37* 7 points

Les résultats d'un vote ayant trois issues possibles 'A', 'B' et 'C' sont stockés dans un tableau.

Exemple :

```
Urne = ['A', 'A', 'A', 'B', 'C', 'B', 'C', 'B', 'C', 'B']
```

La fonction `depouille` doit permettre de compter le nombre de votes exprimés pour chacune des issues. Elle prend en paramètre un tableau et renvoie le résultat dans un dictionnaire dont les clés sont les noms des issues et les valeurs le nombre de votes en leur faveur.

La fonction `vainqueur` doit désigner le nom du ou des gagnants. Elle prend en paramètre un dictionnaire dont la structure est celle du dictionnaire renvoyé par la fonction `depouille` et renvoie un tableau. Ce tableau peut donc contenir plusieurs éléments s'il y a des ex-aequo.

Compléter les fonctions `depouille` et `vainqueur` ci-après pour qu'elles renvoient les résultats attendus.

```
def depouille(urne):
    resultat = ...
    for bulletin in urne:
        if ...:
            resultat[bulletin] = resultat[bulletin] + 1
        else:
            ...
    return resultat


def vainqueur(election):
    vainqueur = ''
    nmax = 0

    for candidat in election:
        if ... > ... :
            nmax = ...
            vainqueur = candidat
    liste_finale = [nom for nom in election if election[nom] == ...]
    return ...
```

Exemples d'utilisation :

```
>>> election = depouille(urne)
>>> election
{'B': 4, 'A': 3, 'A': 3}

>>> vainqueur(election)
['B']
```