

Contrôle de connaissances

Nom : Prénom :

Exercice 1 : (10 points)

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 : • <code>creer_pile_vide</code> : $\emptyset \rightarrow Pile$ <code>creer_pile_vide()</code> renvoie une pile vide • <code>est_vide</code> : $Pile \rightarrow Booleen$ <code>est_vide()</code> envoie <code>True</code> si pile est vide, <code>False</code> sinon • <code>empiler</code> : $Pile, Element \rightarrow \emptyset$ <code>empiler(pile, element)</code> ajoute <code>element</code> à la pile <code>pile</code> • <code>depiler</code> : $Pile \rightarrow element$ <code>depiler(pile)</code> envoie l'élément au sommet de la pile en le retirant de la pile

Structure de données abstraite : File
Utilise : Élément, Booléen
Opérations : • <code>creer_file_vide</code> : $\emptyset \rightarrow File$ <code>creer_file_vide()</code> renvoie une file vide • <code>est_vide</code> : $File \rightarrow Booleen$ <code>est_vide()</code> envoie <code>True</code> si file est vide, <code>False</code> sinon • <code>enfiler</code> : $File, Element \rightarrow \emptyset$ <code>enfiler(file, element)</code> ajoute <code>element</code> à la file <code>file</code> • <code>defiler</code> : $File \rightarrow element$ <code>defiler(file)</code> envoie l'élément au sommet de la file en le retirant de la file

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

enfilement $\rightarrow$ "rouge" "vert" "jaune" "rouge" "jaune" $\rightarrow$ defilement

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.

[illegible]

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 $F = \frac{\text{"rouge" "vert" "jaune" "rouge" "jaune"}}{}$ alors l'appel `former_pile(F)` va renvoyer la pile P ci-dessous :

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

This image shows a single sheet of white paper with horizontal blue or grey ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

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 : (10 points)

Cet exercice porte sur les structures de données (files et la programmation objet en langage python)

Un supermarché met en place un système de passage automatique en caisse.

Un client scanne les articles à l'aide d'un scanner de code-barres au fur et à mesure qu'il les ajoute dans son panier.

Les articles s'enregistrent alors dans une structure de données.

La structure de données utilisée est une file définie par la classe `Panier`, avec les primitives habituelles sur la structure de file.

Pour faciliter la lecture, le code de la classe `Panier` n'est pas écrit.

```
class Panier:
    def __init__(self):
        """Initialise la file comme une file vide."""
    def est_vide(self):
        """Renvoie True si la file est vide, False sinon."""
    def enfiler(self, e):
        """Ajoute l'element e derniere position de la file,
        ne renvoie rien."""
    def defiler(self):
        """Retire le premier element de la file et le renvoie."""
```

Le panier d'un client sera représenté par une file contenant les articles scannés.

Les articles sont représentés par des tuples (`code_barre`, `designation`, `prix`, `horaire_scan`) où :

- `code_barre` est un nombre entier identifiant l'article ;
- `designation` est une chaine de caractères qui pourra être affichée sur le ticket de caisse ;
- `prix` est un nombre décimal donnant le prix d'une unité de cet article ;
- `horaire_scan` est un nombre entier de secondes permettant de connaître l'heure où l'article a été scanné.

1. On souhaite ajouter un article dont le tuple est le suivant

(31002, "café noir", 1.50, 50525).

Écrire le code utilisant une des quatre méthodes ci-dessus permettant d'ajouter l'article à l'objet de classe `Panier` appelé `panier1`.

.....
.....
.....
.....
.....

2. On souhaite définir une méthode `remplir(panier_temp)` dans la classe `Panier` permettant de remplir la file avec tout le contenu d'un autre panier `panier_temp` qui est un objet de type `Panier`.

Recopier et compléter le code de la méthode `remplir` en remplaçant chaque `_____` par la primitive de file qui convient.

```
def remplir(self, panier_temp):  
    while not panier_temp._____  
        article = panier_temp._____  
        self. _____(article)
```

3. Pour que le client puisse connaître à tout moment le montant de son panier, on souhaite ajouter une méthode `prix_total()` à la classe `Panier` qui renvoie la somme des prix de tous les articles présents dans le panier.
Écrire le code de la méthode `prix_total`. Attention, après l'appel de cette méthode, le panier devra toujours contenir ses articles.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

4. Le magasin souhaite connaître pour chaque client la durée des achats. Cette durée sera obtenue en faisant la différence entre le champ `horaire_scan` du dernier article scanné et le champ `horaire_scan` du premier article scanné dans le panier du client. Un panier vide renverra une durée égale à zéro. On pourra accepter que le panier soit vide après l'appel de cette méthode.
Écrire une méthode `duree_courses` de la classe `Panier` qui renvoie cette durée.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....