Nom, prénom :

Évaluation NSI – Pile File

On crée une classe `Pile` qui modélise la structure d'une pile d'entiers. Le constructeur de la classe initialise une pile vide. La définition de cette classe sans l'implémentation de ses méthodes est donnée ci-dessous.

```
class Pile:
    def __init__(self):
        """Initialise la pile comme une pile vide."""

    def est_vide(self):
        """Renvoie True si la liste est vide, False sinon."""

    def empiler(self, e):
        """Ajoute l'élément e sur le sommet de la pile, ne renvoie
        rien."""

    def depiler(self):
        """Retire l'élément au sommet de la pile et le renvoie."""

    def nb_elements(self):
        """Renvoie le nombre d'éléments de la pile. """

    def afficher(self):
        """Affiche de gauche à droite les éléments de la pile, du
        fond de la pile vers son sommet. Le sommet est alors
        l'élément affiché le plus à droite. Les éléments sont
        séparés par une virgule. Si la pile est vide la méthode
        affiche « pile vide »."""
```

Seules les méthodes de la classe ci-dessus doivent être utilisées pour manipuler les objets `Pile`.

Question 1 :

a) Écrire une suite d'instructions permettant de créer une instance de la classe `Pile` affectée à une variable `pile1` contenant les éléments 7, 5 et 2 insérés dans cet ordre.
Ainsi, à l'issue de ces instructions, l'instruction `pile1.afficher()` produit l'affichage : 7, 5, 2.

```
1.  pile1 = Pile()
2.  pile1.empiler(7)
3.  pile1.empiler(5)
4.  pile1.empiler(2)
```

b) Donner l'affichage produit après l'exécution des instructions suivantes.

```
element1 = pile1.depiler() # 2  # fond - 7 - 5 - sommet
pile1.empiler(5)                # fond - 7 - 5 - 5 - sommet
pile1.empiler(element1)         # fond - 7 - 5 - 5 - 2 - sommet
pile1.afficher()                # 7, 5, 5, 2
```

Question 2 :

On donne la fonction `mystere` suivante :

```
def mystere(pile, element):
    pile2 = Pile()
    nb_elements = pile.nb_elements()
    for i in range(nb_elements):
        elem = pile.depiler()
        pile2.empiler(elem)
        if elem == element:
            return pile2
return pile2
```

a) Dans chacun des quatre cas suivants, quel est l'affichage obtenu dans la console ?

• Cas n°1

```
>>>pile.afficher()
7, 5, 2, 3
>>>mystere(pile, 2).afficher()
3, 2
```

• Cas n°2

```
>>>pile.afficher()
7, 5, 2, 3
>>>mystere(pile, 9).afficher()
3, 2, 5, 7
```

• Cas n°3

```
>>>pile.afficher()
7, 5, 2, 3
>>>mystere(pile, 3).afficher()
3
```

• Cas n°4

```
>>>pile.est_vide()
True
>>>mystere(pile, 3).afficher()
pile vide
```

b) Expliquer ce que permet d'obtenir la fonction `mystere`.

Dépile dans une nouvelle pile les éléments de *pile* jusqu'à que celle-ci soit vide et retourne la nouvelle pile. Si un élément dépilé est égal à *element*, la fonction s'arrête prématurément et renvoie la nouvelle pile contenant les éléments dépilés jusqu'à présent.

Question 3 :

Écrire une fonction `etendre(pile1, pile2)` qui prend en arguments deux objets `Pile` appelés `pile1` et `pile2` et qui modifie `pile1` en lui ajoutant les éléments de `pile2` rangés dans l'ordre inverse. Cette fonction ne renvoie rien.

On donne ci-dessous les résultats attendus pour certaines instructions.

```
>>>pile1.afficher()
7, 5, 2, 3
>>>pile2.afficher()
1, 3, 4
>>>etendre(pile1, pile2)
>>>pile1.afficher()
7, 5, 2, 3, 4, 3, 1
>>>pile2.est_vide()
True
```

```
# Boucle for reprise du code de la fonction mystere
def etendre(pile1, pile2):
    nb_elements = pile2.nb_elements()
    for i in range(nb_elements):
        pile1.empiler(pile2.depiler())

# Ou avec une boucle while ...
def etendre(pile1, pile2):
    while not pile2.est_vide():
        pile1.empiler(pile2.depiler())
```

Question 4 :

Écrire une fonction `supprime_toutes_occurences(pile, element)` qui prend en arguments un objet `Pile` appelé `pile` et un élément `element` et supprime tous les éléments `element` de `pile`.

On donne ci-dessous les résultats attendus pour certaines instructions.

```
>>>pile.afficher()
7, 5, 2, 3, 5
>>>supprime_toutes_occurences (pile, 5)
>>>pile.afficher()
7, 2, 3
```

```
# On reprend quasiment tout le code de la fonction mystere ...
def supprime_toutes_occurences(pile, element):
    pile2 = Pile()
    nb_elements = pile.nb_elements()
    for i in range(nb_elements):
        elem = pile.depiler()
        if elem != element:
            pile2.empiler(elem)
     etendre(pile, pile2)        # On réutilise la fonction etendre
```