Implémentation de listes

N.S.I.

1. Listes

Pour rappel, la liste des actions pouvant être réalisées sur une liste.

Actions	Instructions
Créer une liste **L** vide	`L = créer_liste_vide()`
Tester si la liste **L** est vide	`liste_vide(L)`
Ajouter un élément **x** en tête de la liste **L**	`ajoute_en_tete(x, L)`
Supprimer la tête **x** d'une liste **L** et renvoyer cette tête **x**	`suppr_en_tete(L)`
Compter le nombre d'éléments dans une liste **L**	`compte(L)`
Créer une nouvelle liste **L1** à partir d'un élément **x** et d'une liste existante **L**	`L1 = cons(x, L)`

2. Implémentation suivant le paradigme impératif

✍ Compléter l'implémentation des fonctions correspondant aux actions réalisables sur une liste, en respectant le paradigme impératif.

```
def creer_liste_vide():
    """retourne une liste vide"""
    return []

def liste_vide(liste):
    """teste si la liste liste est vide"""


def ajout(element, liste):
    """ajoute un élément dans la liste"""


def suppr(liste):
    """supprime l'élément en tête de liste et renvoie cet élément"""


def compte(liste):
    """renvoie le nombre d'élément constituant la liste liste"""


def cons(element, liste):
    """retourne une nouvelle liste à partir d'un élément element et d'une liste liste"""
```

3. Implémentation suivant le paradigme objet

✍ Compléter l'implémentation des fonctions correspondant aux actions réalisables sur une liste, en respectant le paradigme objet. Inclure dans la classe **`Liste`**, une méthode permettant d'afficher la liste (**`get_liste()`**).

```
class Liste:
    def __init__(self): # constructeur
        self.liste = []

    def liste_vide(self): # accesseur
        """ teste si la liste est vide"""
```

```
    def ajout(self, element): # mutateur
        """ajoute un élément à la liste"""


    def suppr(self): # mutateur
        """supprime la tête de la liste et renvoie cet élément"""


    def compte(self): # accesseur
        """retourne le nombre d'élément constituant la liste"""


    def cons(self, element): # mutateur
        """construit une nouvelle liste en ajoutant element"""


    def get_liste(self): # accesseur
        """retourne le contenu de la liste"""
```

4. Implémentation suivant le paradigme fonctionnel

✍ Compléter l'implémentation des fonctions correspondant aux actions réalisables sur une liste, en respectant le paradigme fonctionnel.

```
def creer_liste_vide():
    """retourne une liste vide"""


def liste_vide(liste):
    """teste si la liste est vide"""


def ajout(element, liste):
    """ajoute un élément dans la liste et retourne la nouvelle liste"""


def suppr(liste):
    """retourne l'élément en tête de liste, supprime cet élément de la liste et retourne la
nouvelle liste"""


def compte(liste):
    """renvoie le nombre d'élément constituant la liste"""


def cons(element, liste):
    """retourne une nouvelle liste crée à partir d'un élément et d'une liste"""
```