1. Paradigme fonctionnel ou pas ?

✍ Les fonctions suivantes sont-elles conformes au paradigme fonctionnel ? Si ce n'est pas le cas, les modifier pour les rendre conformes.

1.1. Exemple 1

```
def incremente():
    return x + 1

x = 5
print(incremente())
```

1.2. Exemple 2

```
def empile(ma_pile, valeur):
    ma_pile.append(valeur)
    return ma_pile

print(empile([], 5))
```

1.3. Exemple 3

```
def concatene(l1, l2):
    return l1 + l2

print(concatene([1, 2], [3, 4]))
```

1.4. Exemple 4

```
def concatene(l1, l2):
    l1 += l2
    return l1

print(concatene([1, 2], [3, 4]))
```

2. λ-calcul

✍ À l'aide du lambda calcul, implémenter une fonction :

- Qui ajoute 10 à un paramètre

```
f1 = . . . . . .
>>> f1(5)
15
```

- Qui multiplie deux paramètres

```
f2 = . . . . . .
>>> f2(3, 4)
12
```

- Qui reçoit un tuple et qui renvoie que le premier terme

```
filtre_rouge = . . . . . .
>>> tuple_couleur = (50, 255, 20)
>>> filtre_rouge(tuple_couleur)
(50, 0, 0)
```

3. Longueur d'une liste

La fonction **size** renvoie la taille d'une liste.

```
def size(liste):
    s = 0
    for x in liste:
        s += 1
    return s

>>> liste = [0, 2, 3, 8, 9]
>>> size(liste)
5
```

✍ Pourquoi cette fonction ne répond pas au paradigme fonctionnel ?

✍ Compléter la fonction **size** pour la rendre conforme au paradigme fonctionnel.

```
def size(liste):
    if not liste:
        return 0
    else:
        return . . . . . .
```

4. Somme et multiplication

✍ Compléter l'implémentation de la fonction **calcul** qui prend en paramètre une fonction pour calculer soit la somme soit le produit des éléments d'une liste.

```
tab = [10, 20, 30]

som = lambda x, y: x + y
mul = lambda x,y: x * y

def calcul(fonct, liste):
    if len(liste) == 1:
        return liste[0]
    else:
        return . . . . . .

>>> calcul(som, tab)
60
>>> calcul(mul, tab)
6000
```