Informatique PCSI
Corrigé TP 1

1 Fonctions de recherche à compléter

```
def recherche2(tab, val):
    i = 0
    while i < len(tab)-1 and tab[i] != val:
        i = i + 1
    return tab[i] == val
```

```
def recherche3(tab, val):
    for i in range(len(tab)):
        if tab[i] == val:
            return True
    return False
```

```
def recherche4(tab, val):
    for elt in tab:
        if elt == val:
            return True
    return False
```

```
def recherche7(tab, val):
    for indice in range(len(tab)):
        if tab[indice] == val:
            return indice
    return len(tab) # ou indice + 1
```

```
def recherche9(tab, val):
    for i in range(len(tab)):
        indice = len(tab) - 1 - i
        if tab[indice] == val:
            return indice
    return len(tab)
```

2 Exercices

2.1 Exercice 1

La variable `i` prend successivement les valeurs 2, 3 et 4. Au premier passage dans la boucle `i` vaut 2 donc `n` prend la valeur 2. Au deuxième passage `i` vaut 3 et `n` prend la valeur 6. Au troisième passage `i` vaut 4 et `n` prend la valeur 24. Les valeurs finales de `i` et `n` sont respectivement 4 et 24.

2.2 Exercice 2

Une solution est de parcourir le conteneur en comptant les éléments un par un.

```
def longueur(conteneur):
    """conteneur est du type str ou list
    renvoie le nombre d'éléments du conteneur"""
    cpt = 0
    for elt in conteneur:
        cpt = cpt + 1
    return cpt
```

2.3 Exercice 3

```
def pair_impair(uplet):
    """uplet est un tuple contenant des entiers,
    renvoie la liste des entiers pairs et celle des entiers impairs"""
    liste_pair = []
    liste_impair = []
    for n in uplet:
        if n % 2 == 0:
            liste_pair.append(n)
        else:
            liste_impair.append(n)
    return liste_pair, liste_impair
```

2.4 Exercice 4

```
def produit(n, nombres):
    """n est de type int et nombres de type list
    multiplie chaque élément de nombres par n
    et renvoie la nouvelle liste"""
    liste = []
    for x in nombres:
        liste.append(x * n)
    return liste

# ou plus simplement en compréhension
def produit(n, nombres):
    return [n * x for x in nombres]
```

2.5 Exercice 5

1. Il faut commencer par calculer la moyenne avec une boucle sur les éléments de la liste. Ensuite à l'aide d'une seconde boucle, on calcule les écarts dont on fait la somme.

```
def distance(liste):
    somme = 0
    for x in liste:
        somme = somme + x
    m = somme / len(liste)
    ecarts = 0
    for x in liste:
        ecarts = ecarts + abs(x - m)
    return ecarts
```

2. Le coût de la deuxième boucle est linéaire en la taille de la liste. Donc le coût du programme est similaire à celui du calcul de la moyenne.

2.6 Exercice 6

La variable `k` prend la valeur 2. La valeur finale de `j` est 0 et `liste[0]` est le premier élément de la liste, soit 5, qui est alors échangé avec 2. On obtient la liste `[2, 8, 12, 13, 17, 5]`.