Informatique PCSI
TP 1 : parcours séquentiel

1 Fonctions de recherche à compléter

- **Recherche de la présence d'une valeur avec une boucle `while`**

```
def recherche1(tab, val):
    presence = False
    i = 0
    while i < len(tab) and not presence:
        if tab[i] == val:
            presence = True
        i = i + 1
    return presence
```

Que vaut `recherche1([1, 2, 3, 4], 3)` ?
Que vaut `recherche1([1, 2, 3, 4], 5)` ?

Compléter le fonction `recherche2` qui doit produire les mêmes résultats que `recherche1` :

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

Compléter les deux fonctions `recherche3` et `recherche4` qui produisent les mêmes effets.
Avec une boucle `for` et une sortie de boucle anticipée :

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

Une fonction semblable à la précédente mais qui n'utilise pas les indices :

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

Avec la fonction `recherche5` qui suit, c'est Python qui fait tout le travail.

```
def recherche5(tab, val):
    return val in tab
```

- **Recherche de la première occurrence**
  Avec une boucle `while` :

```
def recherche6(tab, val):
    indice = 0
    while indice < len(tab) and tab[indice] != val:
        indice = indice + 1
    return indice # renvoie len(tab) si échec
```

Compléter la fonction avec une boucle `for` et une sortie de boucle anticipée :

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

- **Recherche de la dernière occurrence**
  Pour la recherche de la dernière occurrence, on parcourt tout le tableau avec une boucle `for` :

```
def recherche8(tab, val):
    indice = len(tab)
    for i in range(len(tab)):
        if tab[i] == val:
            indice = i
    return indice
```

On pourrait aussi chercher la première occurrence à partir de la fin. Avec une petite modification de la fonction `recherche7`, on obtient la fonction `recherche9` ci-dessous à compléter :

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

2 Exercices

2.1 Exercice 1

Quelles sont les valeurs finales de `i` et `n` après le code qui suit ?

```
n = 1
for i in range(2, 5):
    n = n * i
```

2.2 Exercice 2

Écrire une fonction `longueur` qui prend en paramètre une chaîne de caractères ou une liste et renvoie la longueur de la chaîne ou de la liste. Il est interdit d'utiliser la fonction `len`.

Il est nécessaire de parcourir le conteneur, chaîne ou liste.

2.3 Exercice 3

Écrire une fonction qui prend en argument un n-uplet composé d'entiers et renvoie un couple de deux listes : la première contient les nombres pairs et la seconde les nombres impairs.

2.4 Exercice 4

Écrire une fonction `produit` qui prend en paramètres un entier naturel `n` et une liste de nombres `nombres` et renvoie une nouvelle liste obtenue en multipliant chaque élément de la liste `nombres` par `n`.

2.5 Exercice 5

Calcul d'écarts

1. Écrire une fonction `distance` qui prend en argument une liste de nombres, calcule les écarts en valeur absolue entre chaque nombre de la liste et la moyenne de ces nombres, et renvoie la somme des écarts. (En Python la fonction `abs` renvoie la valeur absolue du nombre donné en paramètre).
2. Quel est le coût de ce programme ?

2.6 Exercice 6

On considère la liste `liste = [5, 8, 12, 13, 17, 2]` et le code qui suit :

```
k = liste[5]
j = 4
while j > 0 and liste[j] > k:
    j = j - 1
liste[j], liste[5] = liste[5], liste[j]
```

Obtient-on une erreur et sinon quel est l'état final de la liste ?