# $2I003$ TD 2 Terminaison et validité d'un algorithme itératif

## Exercice 1 – Factorielle itérative

On considère la fonction `FactorielleIt` définie de la manière suivante pour calculer la factorielle pour $n \in \mathbb{N}$ :

```
def FactorielleIt(n):
    i=1; tmp=1
    while i<=n:
        tmp=tmp*i
        i=i+1
    return tmp
```

### Question 1
Montrer la terminaison de l'algorithme.

### Question 2
Soit $tmp_i$ pour $i \in \{1, \cdots, n\}$ la valeur de la variable $tmp$ aux instants suivants :
- $tmp_1 = 1$ est la valeur de $tmp$ juste avant d'entrer dans la boucle ;
- pour $i \in \{2, \cdots, n+1\}$, $tmp_i$ est la valeur de $tmp$ à la fin du corps de boucle (juste après l'incrémentation de $i$).

Que vaut $tmp_i$ en fonction de $i$ ? En déduire un invariant de boucle et démontrer le.

### Question 3
En déduire la validité de la fonction `FactorielleIt`.

## Exercice 2 – Somme des éléments d'un tableau

On considère la fonction `Somme` définie de la manière suivante pour sommer les éléments d'un tableau :

```
def Somme(tab):
    tmp = 0
    i = 0
    NbElem = len(tab)
    while i < NbElem :
        tmp = tmp + tab[i]
        i = i + 1
    return tmp
```

### Question 1
Montrez la terminaison de l'algorithme.

### Question 2
Soit $tmp_i$ pour $i \in \{0, \cdots, NbElem\}$ la valeur de la variable $tmp$ aux instants suivants :

— $tmp_0$ est la valeur de $tmp$ juste avant d'entrer dans la boucle ;
— pour $i \in \{1, \cdots, NbElem\}$, $tmp_i$ est la valeur de $tmp$ à la fin du corps de boucle (juste après l'incrémentation de $i$). Ainsi, $tmp_1$ vaut $tab[0]$.

Que vaut $tmp_i$ en fonction de $i$ ? Le démontrer par récurrence sur $i$.

### Question 3
En déduire la validité de la fonction `Somme`.

## Exercice 3 – Recherche séquentielle d'un élément dans un tableau non trié

Soit $tab$ un tableau de $n$ entiers et $elem$ un entier. Le but est d'étudier deux fonctions qui retournent l'indice $i$ minimum tel que $tab[i] = elem$.

On suppose tout dabord que $tab$ n'est pas triée et que $elem$ est dans $tab$. On considère alors la fonction itérative suivante :

```
def Recherche(elem, tab):
    i = 0
    while elem!=tab[i] :
        i = i + 1
    return i
```

### Question 1
Montrez la terminaison de la fonction `Recherche`.

### Question 2
A la fin du corps de boucle (juste après l'incrémentation de $i$), que peut-on dire de $elem$ et des éléments de $tab[0 \ldots i-1]$ ? En déduire un invariant de boucle et démontrer le.

### Question 3
En déduire la validité de la fonction `Recherche`.

## Exercice 4 – Recherche séquentielle d'un élément dans un tableau trié

On suppose que $tab$ est un tableau trié en ordre croissant et que $elem$ n'est pas forcément un élément de $tab$. On considère alotrs la fonction itérative suivante :

```
def RechercheTrie(elem, tab):
    i = 0
    n=len(tab)
    while i<n and elem > tab[i] :
        i = i + 1
    if i<n  and elem==tab[i] :
        return i
    return -1
```

### Question 1
Montrez la terminaison de la fonction `RechercheTrie`.

### Question 2
Exprimer un invariant de boucle à la fin du corps de boucle et le démontrer par récurrence.

### Question 3
Etudier la sortie de boucle pour en déduire la validité de la fonction.

## Exercice 5 – Pousser le plus grand élément d'un tableau

On étudie dans cet exercice la fonction itérative Push dont le code suit :

```
def Push(tab,k):
      j=1
      while(j<k):
            if tab[j]<tab[j-1]:
                  tmp=tab[j-1]; tab[j-1]=tab[j]; tab[j]=tmp
            j=j+1
            print("j=",j,"  tab=",tab)
```

On suppose que si $tab$ est un tableau de $n$ éléments, $k \in \{1, \ldots, n\}$.

### Question 1
Exécuter Push (tab,6) pour le tableau $tab[0 \cdots 7] = [8, 7, 12, 5, 15, 4, 3, 9]$. Que fait cette fonction ?

### Question 2
Montrez que l'appel `Push(tab,k)` se termine.

### Question 3
Soit $tab_1 = tab$ et pour $j \in \{2, \cdots, k\}$, $tab_j$ désigne le tableau $tab$ obtenue à la fin du corps de boucle de `Push` (après l'incrémentation de $j$).

Exprimer l'invariant de boucle en fonction de la suite de tableaux $tab_i$ puis le démontrer.

### Question 4
Démontrez que, la fonction `Push(tab,k)` pour $k \in \{1, \cdots, n\}$ a réorganisé les éléments du tableau $tab$ de sorte que :

1. $tab[k-1]$ contient le plus grand élément de $tab[0 \cdots k-1]$ ;
2. Les éléments de $tab[k \cdots n-1]$ n'ont pas été modifiés par `Push(tab,k)`.

## Exercice 6 – Tri à bulles

On considère l'algorithme de tri à bulles suivant :

```
def BubleSort(tab):
      i=0
      n=len(tab)
      while i<n :
            Push(tab, n-i)
            i=i+1
            print("i=", i, "  tab=", tab)
```

### Question 1
Appliquer cette fonction de tri au tableau de $n = 6$ éléments donné par $t[0 \cdots 5] = [18, 17, 4, 12, 1, 2]$. On ne considère que l'affichage de la fonction BubleSort (on oublie l'affichage de la fonction Push définie dans l'exercice précédent).

### Question 2
Montrer que `BubleSort(tab)` se termine.

### Question 3

Soit la suite $tab_0^\star = tab$ et pour tout $i \in \{1, \cdots, n\}$, $tab_i^\star$ est le tableau $tab$ à la fin du corps de boucle de la fonction. Démontrez la propriété $\Pi^\star(i)$, $i \in \{1, \cdots, n\}$ :

1. Le sous tableau $tab_i^\star[0 \cdots n-i-1]$ contient les mêmes éléments que $tab_{i-1}^\star[0 \cdots n-i-1]$ ;
2. Le sous tableau $tab_i^\star[n-i \cdots n-1]$ contient les $i$ plus grands éléments de $tab$ rangés en ordre croissant.

### Question 4

En déduire que `BubleSort(tab)` trie les $n$ éléments de $tab$ en ordre croissant.

## Exercice 7 – Miroir d'un tableau

Le but de cet exercice est d'étudier un algorithme qui inverse les éléments d'un tableau sans utiliser une structure supplémentaire. La fonction `swapp` permet d'inverser les éléments $t[i]$ et $t[j]$. La fonction `miroir` inverse les éléments d'un tableau. Le code de ces fonctions suit. L'appel `len(tab)` renvoie le nombre d'éléments du tableau $tab$. L'instruction `n%2` renvoie la valeur de $n$ modulo 2.

```
def swapp (tab, i, j):
    aux = tab[i]; tab[i]=tab[j]; tab[j]=aux

def miroir (tab):
    n = len(tab)
    j = n/2
    if (n%2 == 0):    # Si n est pair
        i = n/2 - 1
    else:
        i = n/2
    while (j<n):
        swapp(tab, i, j)
        i = i -1
        j = j +1
        print tab
```

### Question 1

Exécutez la fonction `miroir` pour les tableaux $tab = [2, 7, 9, 3, 1]$ et $tab = [7, 9, 2, 4, 3, 10]$.

### Question 2

Calculer le nombre exact d'itérations de la boucle principale de la fonction `miroir`. En déduire la terminaison de la fonction `miroir` pour tout tableau tab.

Par la suite, on pose $k^\star = n - \lfloor \frac{n}{2} \rfloor$. Pour toute valeur $k \in \{0, \cdots, k^\star\}$, on note $tab_k$ le tableau $tab$ à la fin de la $k$-ième itération (juste après avoir incrémenté $j$), $i_k$ la valeur correspondante de $i$ et $j_k$ celle de $j$. Les valeurs $tab_0$, $i_0$ et $j_0$ correspondent aux valeurs de respectivement $tab$, $i$ et $j$ juste avant d'entrer dans la boucle.

### Question 3

Démontrer par récurrence sur $k \in \{0, \cdots, k^\star\}$, les propriétés suivantes :

1. $i_k = i_0 - k$ et $j_k = j_0 + k$ ;
2. $tab_k[0 \cdots i_k] = tab_0[0 \cdots i_k]$ et $tab_k[j_k \cdots n] = tab_0[j_k \cdots n]$ ;
3. $tab_k[i_k + 1 \cdots j_k - 1]$ est le miroir de $tab_0[i_k + 1 \cdots j_k - 1]$.

### Question 4

Démontrez que $i_{k^\star} = -1$ et $j_{k^\star} = n$. En déduire la validité de la fonction `miroir`.