

	Épreuve pratique	NSI T <sup>ale</sup>
	Entraînement numéro 1	TP

\*

## 1. EXERCICE 1 (4 points)

Programmer la fonction `recherche`, prenant en paramètre un tableau non vide `tab` (type `list`) d'entiers et un entier `n`, et qui renvoie l'indice de la **dernière** occurrence de l'élément cherché. Si l'élément n'est pas présent, la fonction renvoie la longueur du tableau.

Exemples :

```
>>> recherche ([ 5, 3 ], 1)
2
>>> recherche ([ 2, 4 ], 2)
0
>>> recherche ([ 2, 3, 5, 2, 4 ], 2)
3
```

## 2. EXERCICE 2 (4 points)

On souhaite programmer une fonction donnant la distance la plus courte entre un point de départ et une liste de points. Les points sont tous à coordonnées entières.

Les points sont donnés sous la forme d'un tuple de deux entiers.

La liste des points à traiter est donc un tableau de tuples.

On rappelle que la distance entre deux points du plan de coordonnées  $(x ; y)$  et  $(x' ; y')$  est donnée par la formule :

$$d = \sqrt{(x - x')^2 + (y - y')^2}$$

On importe pour cela la fonction racine carrée (`sqrt`) du module `math` de Python.

On dispose d'une fonction `distance` et d'une fonction `plus_courte_distance` :

```
from math import sqrt # import de la fonction racine carrée

def distance(point1, point2):
    """ Calcule et renvoie la distance entre deux points. """
    return sqrt((...) ** 2 + (...) ** 2)

assert distance((1, 0), (5, 3)) == 5.0, "erreur de calcul"

def plus_courte_distance(tab, depart):
    """ Renvoie le point du tableau tab se trouvant à la plus
    courte distance du point depart. """
    point = tab[0]
    min_dist = ...
    for i in range(1, ...):
        if distance(tab[i], depart) ...:
            point = ...
            min_dist = ...
    return point

assert plus_courte_distance([(7, 9), (2, 5), (5, 2)], (0, 0)) == (2,
5), "erreur"
```

Recopier sous Python ces deux fonctions puis compléter leur code et ajouter une ou des déclarations (`assert`) à la fonction `distance` permettant de vérifier la ou les préconditions.