RECURSION

EXERCICE 1

Ecrire un programme qui permet de saisir un entier **n** > 0, de déterminer et afficher le nombre de ses chiffres.

Exemple :
Si n = 165, le programme affichera : 165 contains 3 digits
Si n = 2023, le programme affichera : 2023 contains 4 digits

EXERCICE 2

Ecrire un programme qui permet de :

- Saisir un entier **n** avec **4 ≤ n ≤ 100**.
- Remplir un Tableau **A** par les noms de **n** élèves.
- Afficher le tableau **A** (solution récursive)

NB :

Trouver 2 solutions récursives pour afficher le tableau **A** :

- La première permet d'afficher le contenu du tableau en commençant par la première case.
- La deuxième permet d'afficher le contenu du tableau en commençant par la dernière case.

EXERCICE 3

Ecrire un programme qui permet de :

- Générer un entier n avec **5 ≤ n < 90**.
- Remplir un Tableau **A** par **n** entiers.
- Déterminer la plus grande valeur du tableau **A** (solution récursive)
- Afficher cette valeur.

Exemple :
Pour **n** = 6 et

A	20	342	11	68	199	254
	0	**1**	**2**	**3**	**4**	**5**

Le programme affichera : The greatest value in the array is 342

EXERCICE 4

Ecrire un programme qui permet de :

- Saisir une chaine de caractères **st** contenant au moins une lettre majuscule.
- Déterminer la dernière lettre majuscule qui apparaît dans **st** (solution récursive).
- Afficher la lettre.

Exemple :

Si st="Hello WoRld", le programme affichera: `The last capital letter appears in Hello World is R.`

Si st="money Time", le programme affichera: `The last capital letter appears in money Time is T.`

EXERCICE 5

Ecrire un algorithme de la fonction **Parity** qui permet de retourner "Odd" si un entier **n** positif est impair ou "Even" s'il est pair.

On donne : `Fonction Parity(n : entier) : chaine`

EXERCICE 6

Ecrire un algorithme de la fonction **Exist** qui permet de vérifier l'existence d'un caractère **c** dans une chaine de caractères **st**.

On donne : `Fonction Exist(st : chaine ; c : caractère) : Booléen`

EXERCICE 7

Ecrire un algorithme de la fonction **LinearSearch** qui permet de vérifier l'existence d'un entier **m** dans un tableau **A** contenant **n** entiers en utilisant la méthode de recherche séquenciel.

On donne : `Fonction LinearSearch(A : ArrInt ; n , m : entier) : Booléen`

EXERCICE 8

Ecrire un algorithme de la fonction **SumArray** qui permet de calculer la somme des éléments d'un tableau **A** contenant **n** réels.

On donne : `Fonction SumArray(A : ArrFloat ; n : entier) : réel`

EXERCICE 9

Ecrire un algorithme de la fonction **Product** qui permet de calculer le produit de 2 entiers positifs non nuls, **a** et **b**.

On donne : `Fonction Product(a , b : entier) : entier`