Série N°2 (TD & TP)

(Fonctions, Procédures et Récursivité)

Exercice 1

Ecrire l'algorithme qui permet de calculer la combinaison de dénombrement C_N^P. Utiliser une fonction « Fact » qui calcule et retourne la factorielle d'un entier.
Traduire cet algorithme en Python.

Exercice 2

Ecrire un algorithme qui permet de déterminer les 80 premiers nombres premiers ≥ 2. Utiliser une fonction « premier » qui détermine si son argument est premier ou non.
Traduire cet algorithme en Python.

Exercice 3

Ecrire un algorithme qui lit une suite de couples (x,y) terminée par (0,0) et affiche à chaque fois le plus grand diviseur commun des deux nombres. Utiliser une fonction « PGCD » qui rend le plus grand diviseur commun de ses deux paramètres.
Traduire cet algorithme en Python.

Exercice 4

Ecrire une fonction « PPCM » qui permet de déterminer le plus petit commun multiple de deux nombres entiers strictement positifs n et m.
Traduire cet algorithme en Python.

Exercice 5

Ecrire un algorithme qui donne la liste de tous les nombres parfaits compris entre 6 et 10000. Utiliser une fonction « parfait » qui détermine si son argument est parfait ou non.
Un nombre est dit parfait s'il est égal à la somme de tous ses diviseurs stricts (Par exemple 28 est parfait : 28=1+2+4+7+14).
Traduire cet algorithme en Python.

Exercice 6

On considère un ensemble $H_a=\{n\in IN\ /2^n > a\}$; $a\in IN$

1- Ecrire l'algorithme de la fonction MinEnsemble qui permet de déterminer le minimum de l'ensemble H_a : Fonction MinEnsemble (a :entier) :entier
2- Utiliser le résultat de la fonction MinEnsemble pour écrire l'algorithme de la fonction DecimalBinaire qui permet de convertir un entier de la base décimale à la base binaire (le résultat renvoyé est stocké dans un tableau) :
Fonction DecimalBinaire (a :entier) :entier [ ]

Exercice 7

Ecrire un algorithme qui demande à l'utilisateur de saisir un entier N (N ≥2), puis calcule et affiche tous les termes de la suite de Fibonacci, inférieurs ou égaux à N.
La suite de Fibonacci est définie comme suite :

$$\begin{cases} U_0 = 0 \\ U_1 = 1 \\ U_{n+2} = U_{n+1} + U_n \end{cases}$$

1- Utiliser une fonction itérative « fibIter ».
2- Utiliser une fonction récursive « fibRec ».

Traduire cet algorithme en Python.

Exercice 8

Ecrire une fonction récursive permettant de calculer la somme des chiffres d'un entier n positif.
Exemple : Si n = 834, la somme des chiffres de n est 15 (=8+3+4).

Exercice 9

Soit n un entier strictement positif. Écrire une fonction récursive chiffre(n, k) qui permet de retourner le $k^{ième}$ chiffre de n à partir de la droite.

Exemples :

- Le $3^{ième}$ chiffre à partir de la droite de 5739 est 7.
- Le $5^{ième}$e chiffre à partir de la droite de 81467 est 8.

Traduire cet algorithme en Python.

Exercice 10

Écrire une fonction récursive qui calcule :

1- La somme des n premiers entiers naturels (à partir de 1).
2- La somme de deux entiers naturels a et b.
3- Le produit de deux entiers naturels a et b.
4- La puissance de a et b (a étant un réel, et b un entier naturel).
5- Le quotient de deux entiers naturels a par b (b≠0).
6- Le reste de division de a par b (a et b deux entiers naturels, et b≠0).

Traduire cet algorithme en Python.

Exercice 11

Ecrire l'algorithme de la fonction MajoriteCarres suivante :

Fonction MajoriteCarres (T: entier [1..N]) :booleen

La fonction retourne VRAI si le nombre des carres parfaits dans le tableau T est majoritaire.
Traduire cet algorithme en Python.

Exercice 12

Écrire un algorithme qui demande à l'utilisateur de saisir la taille et les éléments d'un tableau d'entiers T. Cet algorithme se sert de deux procédures « afficherMax » et « afficherMin » pour déterminer et afficher le maximum et le minimum des éléments du tableau.
Traduire cet algorithme en Python.

Exercice 13

Ecrire un algorithme qui demande à l'utilisateur de saisir la taille et les éléments d'un tableau d'entiers T, ensuite on demande à l'utilisateur de saisir un entier a. L'objectif étant de vérifier l'existence du nombre a dans T.
Utiliser une fonction « verifierExistence » à deux arguments (T et a) qui retourne l'indice de la première occurrence de a dans T et -1 si a n'existe pas dans T.
Traduire cet algorithme en Python.

Exercice 14

Ecrire une procédure récursive qui permet de réarranger les éléments d'un tableau en ordre inverse.

Procédure OrdreInverse_Recurs(T :entier[1..N], i :entier)

Exercice 15

Soit T un tableau d'entiers de taille n (n≤100).
Ecrire des fonctions récursives pour réaliser les opérations suivantes :

1. Somme : qui permet de retourner la somme des éléments du tableau T.
2. Produit : qui permet de retourner le produit des éléments du tableau T.
3. Moyenne : qui permet de retourner la moyenne des éléments du tableau T.
4. RechElt : qui permet de retourner l'indice de l'élément contenant une valeur donnée
5. RechSeq : qui permet de retourner l'indice d'un élément contenant une valeur donnée dans un vecteur T trié dans l'ordre croissant.
6. NbOcc : qui permet de retourner le nombre d'occurrences d'une valeur donnée dans T.
7. Est_trie : qui permet d'indiquer si le tableau est trié dans l'ordre croissant ou non.