Pour chaque exercice, écrire en plus des sous-programmes demandés un programme principal, et éventuellement des sous-programmes annexes, permettant d'exécuter et de tester vos algorithmes.

Exercice 1 – Sous-programmes récursifs simples

1. Écrire une procédure récursive qui affiche n fois le message **bonjour**.
2. Écrire une fonction récursive qui calcule la somme des n premiers carrés.

Exercice 2 – Suite de Syracuse

Une suite d'entiers dite de Syracuse suit le schéma suivant : on part d'un entier strictement positif ; s'il est pair, on le divise par 2 ; s'il est impair, on le multiplie par 3 et on ajoute 1. Par exemple, à partir de 14, on construit la suite des nombres : 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2… On conjecture que quel que soit le nombre initial, la suite de Syracuse finit toujours par passer par la valeur 1 (puis la suite boucle sur les valeurs 1, 4, 2).

Écrire une procédure récursive permettant d'afficher la suite de Syracuse partant d'un entier donné, en s'arrêtant à la valeur 1.

Exercice 3 – Coefficients binomiaux

Les coefficients binomiaux C(n,k), utilisés par exemple dans le calcul de $(a + b)^n$, peuvent être calculés au moyen de la règle de Pascal : C(n,k) = C(n-1,k-1) + C(n-1,k).

1. En déduire une fonction récursive calculant C(n,k).
2. Écrire une procédure (non récursive) **developpe**, qui prend en paramètre n, et qui affiche la forme développée de $(a+b)^n$.
   Par exemple, pour n = 3, l'affichage devra être a3 + 3a2b + 3ab2 + b3.

Exercice 4 – Suite de Fibonacci

Écrire une fonction **fibo** calculant le n-ième terme de la suite de Fibonacci :

$$\begin{cases} u_0 = 0 \\ u_1 = 1 \\ u_n = u_{n-1} + u_{n-2} \end{cases}$$

Calculer u_n pour $n = 20,\ 30,\ 40,\ 50$. Proposer une version non récursive plus efficace pour le calcul de u_n.

Exercice 5 – Chaines de caractères

Écrire les fonctions récursives effectuant les traitements suivants :

1. Retourner une sous-chaine d'une chaine de caractères, étant données un indice de départ et une longueur.
2. Tester si une chaine est un palindrome.
3. Compter le nombre de d'occurrences d'un caractère dans une chaine.

Exercice 6 – Tri de tableau

T désigne un tableau d'entiers dont les éléments sont triés (par ordre croissant).

1. Écrire un sous-programme récursif permettant de déterminer à quelle position un entier x doit être inséré dans un tableau T.
2. Écrire deux versions (une itérative, une récursive) d'un sous-programme permettant d'insérer un élément x dans un tableau T (afin que celui-ci reste trié).
3. En déduire une version récursive du tri par insertion.
4. Écrire des procédures récursives de saisie, de génération aléatoire et d'affichage d'un tableau (non nécessairement trié).
5. Finaliser le programme afin de tester l'algorithme du tri par insertion.
6. [Question supplémentaire] Écrire un sous-programme de tri utilisant l'algorithme du *tri rapide*.