TP 1-2

1 Suites récurrentes

A. Ecrire un programme qui calcule les 20 premières valeurs de la suite a_n, donnée par : $a_0 = 0$, $a_1 = 1$ et $a_n = 5a_{n-1} - 6a_{n-2}$ pour $n \geq 2$. Comparer a_n avec $3^n - 2^n$.

B. Ecrire un programme qui calcule les 100 premières valeurs de la suite c_n, donnée par : $c_0 = 0$, $c_1 = 1$ et $c_n = c_{n-1} - c_{n-2}$ pour $n \geq 2$. Trouver une formule pour c_n.

C. La population de grenouilles d'un lac quadruple chaque année. Le premier jour de chaque année 100 grenouilles sont déplacées dans un autre lac et initialement il y avaient 100 grenouilles. Soit a_n le nombre de grenouilles apres n années. Ecrire un programme qui affiche les 20 premières valeurs de la suite a_n.

D. Une personne dépose sur un compte à 5% d'intérêt par an la somme de 1000 euros. A partir de la deuxième année la personne dépose chaque année 500 euros de plus sur le compte. Soit e_n la somme disponible sur le compte à la fin de la n-ième année. Ecrire un programme qui affiche les 20 premières valeurs de la suite e_n.

E. Ecrire un programme qui calcule les 20 premières valeurs de la suite f_n, donnée par : $f_0 = 0$, $f_1 = 1$ et $f_n = f_{n-1} + f_{n-2}$ pour $n \geq 2$. Comparer f_n avec $\frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$. Comparer f_n avec $int\left(0.4472135954 \times 1.618033988^n\right)$, où $int(x)$ est l'entier le plus proche de x.

2 Séquences binaires

A. (`Gen_Bin`)
L'ensemble B_n de séquences binaires de longueur n est défini par :

$$B_n = \begin{cases} \emptyset & \text{si } n = 0 \\ 0B_{n-1} \cup 1B_{n-1} & \text{si } n \geq 1. \end{cases} \tag{1}$$

Ecrire un programme, `Gen_Bin` qui affiche, pour un n donné, toutes les séquences binaires de longueur n. Exemple : pour $n = 3$ il affichera 000, 001, 010, 011, 100, 101, 110, 111.

B. (`Gen_Bin_Gray`)

Un *code de Gray* pour l'ensemble B_n est défini par :

$$G_n = \begin{cases} \emptyset & \text{si } n = 0 \\ 0G_{n-1} \cup 1\overline{G}_{n-1} & \text{si } n \geq 1, \end{cases} \tag{2}$$

où $\overline{G}_n$ est le miroir de la liste G_n.
Ecrire un programme, `Gen_Bin_Gray` qui affiche, pour un n donné, le le code de Gray G_n.
Exemple : pour $n = 3$ il affichera 000, 001, 011, 010, 110, 111, 101, 100.

C. (`Gen_Fibo`)
Une *séquence de Fibonacci* est une séquence binaire qui ne contiennent pas deux 1 successifs. Par exemple, les séquences de Fibonacci de longueur 3 sont 000, 001, 010, 100, 101, et on dénote par F_n l'ensemble de séquences de Fibonacci de longueur n. F_n est défini récursivement par :

$$F_n = \begin{cases} \emptyset & \text{si } n = 0 \\ 0, 1 & \text{si } n = 1 \\ 0F_{n-1} \cup 10F_{2-1} & \text{si } n \geq 2. \end{cases} \tag{3}$$

Ecrire un programme, `Gen_Fibo` qui affiche, pour un n donné, l'ensemble F_n.

D. (`Gen_Comb`)
Une séquence binaire de longueur n a le *poids* k si elle a exactement k occurrences de 1. Soit $C_{n,k}$ l'ensemble de suites binaires de longueur n et de poids k. Par exemple, $C_{4,2} = \{0011, 0101, 0110, 1001, 1010, 1100\}$. $C_{n,k}$ est défini récursivement par :

$$C_{n,k} = \begin{cases} \emptyset & \text{si } n = 0 \\ 0C_{n-1,k} & \text{si } k = 0 \text{ et } n \neq 0 \\ 1C_{n-1,k-1} & \text{si } k = n \neq 0 \\ 0C_{n-1,k} \cup 1C_{n-1,k-1} & \text{si } n > k > 0. \end{cases} \tag{4}$$

Ecrire un programme, `Gen_Comb` qui affiche, pour un n et k donné, l'ensemble $C_{n,k}$.

E. (`Gen_Dyck`)
Un *mot de Dyck* est une séquence binaire qui contient autant de 0 que de 1, et chaque préfixe contient au moins autant de 1 que de 0. Par exemple, les mots de Dyck de longueur 6 sont : 101010, 101100, 110010, 110100, 111000, et on dénote par D_n les mots de Dyck de longueur n (nécessairement n est pair). Il est claire que $D_n \subset C_{n,n/2}$, et $D_n = D_{n,n/2}$, où $D_{n,k}$ est défini récursivement par :

$$D_{n,k} = \begin{cases} \emptyset & \text{si } n = 0 \\ 1D_{n-1,k-1} & \text{si } n = 2k \geq 1 \\ 0D_{n-1,k} & \text{si } k = 0 \text{ et } n \geq 1 \\ 0D_{n-1,k} \cup 1D_{n-1,k-1} & \text{si } n > k \geq 1. \end{cases} \tag{5}$$

Ecrire un programme `Gen_Dyck`, qui affiche, pour un n donné, affiche l'ensemble D_n.

3 Combinaisons

Soit $C_{n,k} \subset \{1, 2, 3, \ldots, n\}$ l'ensemble des combinaisons de k parmi n ($1 \leq k \leq n$):

$$C_{n,k} = \{c_1c_2 \ldots c_k \mid c_i < c_j \text{ si } i < j\}.$$

$C_{n,k}$ est défini récursivement par:

$$C(n,k) = \begin{cases} \emptyset & \text{si } n = 0 \\ C(n-1,k) \cup C(n-1,k-1)n & \text{si } n \geq 1 \end{cases} \tag{6}$$

ce qui est la forme *quantitative* de la relation $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$ (triangle de Pascal) avec $card(C(n,k)) = C_n^k$. Par exemple:

$$C_{5,3} = C_{4,3} \cup C_{4,2}\mathbf{5} = \{123, 124, 134, 234\} \cup \{12, 13, 14, 23, 24, 34\}\mathbf{5}$$

$$= \{123, 124, 134, 234, 12\mathbf{5}, 13\mathbf{5}, 14\mathbf{5}, 23\mathbf{5}, 24\mathbf{5}, 34\mathbf{5}\}$$

Écrire un algorithme de génération exhaustive pour $C(n,k)$ en utilisant la **définition récursive**.