Informatique – TP 1
Révisions (structures, suites récurrentes, sommes)

Exercice 1 – Écrire un programme demandant une valeur réelle A et renvoyant le plus petit indice n tel que $\sum_{k=1}^{n} \frac{1}{k} > A$.

Exercice 2 – Soit la suite définie par $u_0 = 0$, et pour tout $n \in \mathbb{N}$, $u_{n+1} = \sqrt{3u_n + 4}$. On montre facilement que cette suite converge vers 4 en croissant.

1. Écrire un programme demandant à l'utilisateur un entier n en renvoyant tous les termes de la suite jusqu'à u_n.
2. Écrire un programme renvoyant le plus petit entier n pour lequel $u_n > 3,99999999$.

Exercice 3 – Calculer $\sum_{n=0}^{1000} u_n$ où $u_0 = 1$ et $\forall n \geqslant 0$, $u_{n+1} = \dfrac{1}{u_n + 1}$.

Exercice 4 –
Soit $F_0 = 0$, $F_1 = 1$ et pour tout $n \in \mathbb{N}$, $F_{n+2} = F_{n+1} + F_n$ (suite de Fibonacci). Écrire une fonction prenant en paramètre une valeur de n et calculant F_n.

Exercice 5 – On définit la suite de Syracuse par $u_0 \in \mathbb{N}^*$, et

$$u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$$

On veut vérifier la propriété suivante : il existe un rang N tel que $u_N = 1$ (et à partir de ce rang, la suite boucle : 4, 2, 1, 4, 2, 1, etc.). Écrire un programme demandant à l'utilisateur une valeur initiale u_0, calculant et affichant les différents termes de la suite tant qu'ils ne sont pas égaux à 1, et affichant pour terminer la première valeur de N pour laquelle $u_N = 1$, ainsi que la plus grande valeur obtenue pour u_n.
Cette propriété, à l'énoncé pourtant très simple, est encore aujourd'hui une conjecture, malgré les efforts acharnés de nombreux mathématiciens plus brillants les uns que les autres.

Exercice 6 – Écrire une fonction prenant en paramètre deux réels strictement positifs a et e, et calculant une valeur approchée de la somme (dont on justifiera la convergence) $\sum_{n=1}^{+\infty} \frac{(-1)^n}{n^a}$ à e près.

Exercice 7 – Écrire une fonction déterminant le minimum des valeurs d'un tableau passé en paramètre

Exercice 8 – Écrire une procédure affichant tous les entiers premiers inférieurs à 32000 (on procèdera à l'aide d'un crible d'Eratosthène).

Exercice 9 – Écrire un programme demandant à l'utilisateur de rentrer une année et affichant à l'écran un booléen :
• `true` si l'année est bissextile
• `false` si l'année n'est pas bissextile.
On rappelle que les années multiples de 4 sont bissextiles, mais les années multiples de 100 ne le sont pas, sauf si elles sont multiples de 400. On rappelle également que ces règles datent de l'avènement du calendrier grégorien en 1582, et qu'avant cette date, toutes les années multiples de 4 étaient bissextiles, sans exception (calendrier julien).

Exercice 10 – Valeur approchée d'une intégrale par la méthode des trapèzes –
Soit a et b deux nombres réels tels que $a < b$ et f une fonction de classe $\mathcal{C}^2$ sur l'intervalle $[a, b]$.

1. Montrer qu'il existe $M \in \mathbb{R}_+^*$ tel que :

$$\left| \int_a^b f(x)\ \mathrm{d}x - \frac{b-a}{2}(f(a) + f(b)) \right| \leqslant M \frac{(b-a)^3}{12}.$$

Indication : on pourra soit intégrer par parties judicieusement (de sorte à garder une symétrie dans le rôle de a et b), soit utiliser une formule de Taylor.

2. Soit $n \in \mathbb{N}^*$. On pose $T_n = \dfrac{b-a}{n}\left(\dfrac{f(a)+f(b)}{2} + \displaystyle\sum_{k=1}^{n-1} f\left(a + k\dfrac{b-a}{n}\right)\right)$.

   (a) En posant $x_k = a + k\dfrac{b-a}{n}$, vérifier que $T_n = \dfrac{1}{2}\displaystyle\sum_{k=0}^{n-1}(x_{k+1} - x_k)(f(x_k) + f(x_{k+1}))$.

   Interpréter T_n géométriquement.

   (b) Démontrer que : $\left|\displaystyle\int_a^b f(x)\ \mathrm{d}x - T_n\right| \leqslant M\dfrac{(b-a)^3}{12n^2}$.

3. (a) Soit f la fonction définie sur $[1,2]$ par $f(t) = \mathrm{e}^{t^2}$. Déterminer un majorant de $|f''|$.

   (b) Écrire une fonction en Pascal prenant en paramètre une marge d'erreur e, et calculant $\displaystyle\int_1^2 \mathrm{e}^{t^2}\ \mathrm{d}t$ à e près.

   (c) Quelle est la valeur de n à retenir pour obtenir une valeur approchée de l'intégrale de la question précédente à 10^{-8} près ?

Exercice 11 – Exponentielle de matrice

Soit B une matrice d'ordre 2. Pour tout entier naturel non nul, et tout réel t, on définit la matrice $E_n(t)$ par :

$$E_n(t) = \sum_{k=0}^{n} \frac{t^k}{k!}B^k, \qquad \text{que l'on note} \qquad E_n(t) = \begin{pmatrix} a_n(t) & c_n(t) \\ b_n(t) & d_n(t) \end{pmatrix}.$$

Écrire un programme en PASCAL demandant à l'utilisateur d'entrer une matrice d'ordre 2 et un entier n, et renvoyant $E_n(t)$. Pour calculer $E_n(t)$, on utilisera un algorithme de type Hörner. On fera attention au fait que l'opération $a_{k-1} + X.A$ sur les polynômes se traduit pour les matrices par $a_{k-1}I_2 + X \cdot A$. On déclarera un type matrice (tableau de taille 2×2), une variable I représentant I_2, une procédure d'entrée d'une matrice par l'utilisateur, et deux fonctions somme et produit faisant la somme de deux matrices et le produit d'une matrice par un scalaire.

Exercice 12 – Génération aléatoire d'une permutation

On se propose dans cet exercice d'écrire un programme permettant de générer aléatoirement une permutation de $[\![1,n]\!]$. *Une permutation* σ *sera représentée par un tableau de taille utile* n*, d'indices variant entre* 1 *et* n*, telles que l'entrée* i *du tableau soit* $\sigma(i)$.
On travaillera dans un tableau suffisammment grand, de taille `Nmax`$= 50$.

Écrire dans un même programme :

1. une procédure mettant à 0 toutes les cases d'un tableau représentant une permutation. Dans la suite, on appellera *case vide* une case dont le contenu est 0 ;
2. une procédure prenant en paramètre l'entier n, taille utile du tableau, deux entiers i et k, et un tableau S, et mettant la valeur k dans la i-ème case **vide** du tableau S ;
3. une procédure prenant en paramètre un entier n et un tableau S, et renvoyant en S une permutation choisie aléatoirement. Pour ce faire, on choisira successivement les images réciproques des éléments $1, \ldots, n$. Ainsi :
   - on choisit d'abord une image réciproque pour 1 : il s'agit d'une valeur aléatoire i dans $[\![1,n]\!]$. Ainsi, $\sigma(i) = 1$, donc la case i doit être mise à 1.
   - on choisit ensuite une image réciproque pour 2 : il nous reste $n-1$ valeurs possibles (i est à exclure) On choisit un nombre j entre 1 et $n-1$, et l'image réciproque correspondra alors au numéro de la j-ième case vide,
   - etc.
4. Une entête et un corps de programme dans lequel on demande à l'utilisateur une valeur de n, et on choisit aléatoirement (sans affichage) une permutation de $[\![1,n]\!]$.
5. **Question subsidiaire :** Écrire une procédure prenant en paramètres d'entrée une permutation S et sa taille, et en paramètre de sortie une permutation T, de sorte que T soit égal, à l'issue de la procédure, à la permutation réciproque de S.

Cette méthode est loin d'être optimale. Une méthode plus rapide consiste à partir d'une permutation donnée, par exemple l'identité, et à faire des échanges de valeurs (composer par des transposition) un certain nombre de fois.