Informatique - Exercices de révision

Exercice 1 : Représentation binaire

1. Donner l'entier représenté par `11101` en binaire et celui représenté par `CA` en hexadécimal.
2. Sachant que $0{,}375 = 0{,}250 + 0{,}125$ et que $0{,}125 = \frac{1}{8}$, écrire la représentation binaire (avec virgule) du nombre décimal 5,375.
3. Écrire une fonction `binaire(n)` qui donne la représentation en binaire de l'entier n sous forme d'une liste de bits.
4. Écrire une fonction `octet(n)` qui renvoie le nombre d'octets nécessaires pour stocker la représentation en binaire de l'entier n.

Exercice 2 : Échantillonnage

On considère un convertisseur analogique numérique dont la plage de tension est 0 - 5V avec une résolution de 16 bits.

1. On mesure une tension en lisant dans Python la valeur numérique donnée par le convertisseur. Quelle est la précision maximale en tension de la mesure ?
2. On échantillonne à 1000 Hz la tension pendant 5s. Quelle est la quantité de mémoire nécessaire ?
3. La liaison entre le convertisseur et l'ordinateur est limitée à 127000 bits par seconde. Est-ce suffisant ?
4. Quelle est la fréquence maximale d'échantillonnage avec une telle liaison ?

Exercice 3 : Questions de cours sur le calcul numérique

1. Écrire une fonction `trapz(f,a,b,n)` qui calcule une valeur approchée de $\int_a^b f(t)\,\mathrm{d}t$ en utilisant la méthode des trapèzes à n pas. Appliquer à la fonction $f(t) = \dfrac{\cos(t^2) - t^3}{1 + t^2}$ sur $[0, \pi]$.
2. Écrire une fonction `dicho(f,a,b)` qui calcule une valeur approchée d'une racine de la fonction f dans $[a, b]$. On suppose que $f(a)f(b) < 0$.
3. On considère la fonction $g(t) = \dfrac{\cos(t) - t^3}{1 + t^2}$. On sait que g a une racine dans l'intervalle $[0, \pi]$.

   Écrire une fonction `Newton(g,x0)` qui calcule une valeur approchée de la racine de g en utilisant la méthode de Newton.

Exercice 4 : Recherche dichotomique

1. Écrire une fonction `rechdicho(L,val)` qui recherche la position de la première occurrence de `val` dans la liste `L` qui est triée dans l'ordre croissant. On écrira l'invariant de boucle avant de programmer.
2. Écrire une version récursive de la fonction précédente. Pour éviter des recopies de listes, on travaillera sur la tranche `L[g:d]` de la liste.

Exercice 5 : SQL

On donne 3 relations **communes**, **departements** et **regions** avec les schémas suivants :

communes (id, dep_id, nom, pop), **departements (id, numero, reg_id, nom)**, **regions (id, nom)**

avec `id` clé primaire (identifiant) de chaque relation, `dep_id` identifiant du département de la commune, `reg_id` identifiant de la région du département, `pop` population de la commune, `nom` nom de la commune ou département ou région, `numero` numéro du département.

Écrire les requêtes nécessaires en langage SQL pour obtenir les informations suivantes :

1. la moyenne des populations des communes de France,
2. les communes de Loire-Atlantique puis celles des "Pays de la Loire",
3. la ou les commune(s) la(es) moins peuplée(s) de France,
4. les 100 communes les plus peuplées (on utilisera `LIMIT 100` qui ne donne que les 100 premiers tuples d'une relation),
5. la moyenne de population des communes pour chaque département,
6. les départements ayant une population totale supérieure à 1 million d'habitants,
7. les départements ayant une population totale supérieure à la moyenne des populations totales des départements,

Exercice 6 : Algorithme de Gauss

Écrire les fonctions nécessaires pour résoudre un système linéaire $AX = B$ de rang maximal avec A matrice carrée de taille n inversible et B matrice colonne à n lignes.

Exercice 7 : Tri fusion

Écrire une fonction **`trifusion(L)`** qui effectue un tri fusion de la liste **`L`**.

Exercice 8 : Manipulation d'échantillons

On suppose que les listes $(t_i)_{i\in[\![0,n-1]\!]}$ et $(y_i)_{i\in[\![0,n-1]\!]}$ sont un échantillonage d'une fonction $\varphi : \forall i \in [\![0, n-1]\!], \quad \varphi(t_i) = y_i$.

1. Écrire une fonction **`valeurMoyenne(Lt, Ly)`** qui prend en argument les listes **`Lt`** et **`Ly`** représentant $(t_i)_{i\in[\![0,n-1]\!]}$ et $(y_i)_{i\in[\![0,n-1]\!]}$ et qui renvoie la valeur moyenne de la fonction φ sur l'intervalle $[t_0, t_{n-1}]$.
2. Écrire une fonction **`ecrire(nom_fichier, Lt, Ly)`** qui prend en argument un nom de fichier, les listes **`Lt`** et **`Ly`** et qui écrit dans le fichier **`nom_fichier`** les valeurs des listes : deux valeurs par ligne séparées par un point virgule « t_i **`;`** y_i**`\n`** ».
3. Écrire une fonction **`lire(nom_fichier)`** qui prend en argument un nom de fichier et qui renvoie les listes **`Lt`** et **`Ly`** lues dans le fichier **`nom_fichier`** sous la forme de deux valeurs par ligne séparées par un point virgule « t_i **`;`** y_i**`\n`** ».
4. Écrire une fonction **`moyenne(p, Lt, Ly)`** qui renvoie une nouvelle liste **`Lz`** dont les valeurs sont les moyennes de p éléments de **`Ly`**, c'est à dire $z_k = \dfrac{1}{p}\displaystyle\sum_{i=0}^{p-1} y_{k+i}$ pour $k \in [\![0, n-p]\!]$. La fonction renverra également la liste **`Ltbis`** qui est la liste **`Lt`** raccourcie.

Exercice 9 : Fonction récursive

On veut écrire une fonction récursive qui renvoie la liste des permutations des entiers de $[\![1, n]\!]$. Une permutation sera représentée par une liste $[a_0, a_1, ..., a_{n-1}]$ de n entiers. On utilisera la méthode suivante :

— La seule permutation des entiers de $[\![1, n]\!]$ pour $n = 1$ est $[1]$.
— si $[a_0, a_1, ..., a_{n-1}]$ est une permutation quelconcque des entiers de $[\![1, n-1]\!]$, alors les permutations recherchées sont $[n, a_0, a_1, ..., a_{n-1}]$, $[a_0, n, a_1, ..., a_{n-1}]$, $[a_0, a_1, n, ..., a_{n-1}]$, ..., $[a_0, a_1, ..., n, a_{n-1}]$, $[a_0, a_1, ..., a_{n-1}, n]$.

Exercice 10 : Manipulation de listes

1. Écrire une fonction **`zip(L1,L2)`** qui prend en argument deux liste de mêmes longueur n et qui renvoie une liste contenant les couples **`[L1[k],L2[k]]`** pour tout entier $k \in [\![1, n]\!]$.
2. Écrire une fonction **`seuil1(L,s)`** qui prend en argument une liste **`L`** et une valeur à comparer **`s`** et qui renvoie l'indice de la première valeur de **`L`** qui est supérieure à **`s`**.
3. Écrire une fonction **`seuil2(L,s)`** qui prend en argument une liste **`L`** et une valeur à comparer **`s`** et qui renvoie l'indice de la dernière valeur de **`L`** qui est supérieure à **`s`**.
4. Écrire une fonction **`seuilx(LP,s)`** qui prend en argument une liste **`LP`** de couples de coordonnées **`[x,y]`** et une valeur à comparer **`s`** et qui renvoie tous les couples donc l'abscisse **`x`** est supérieure au seuil **`s`**.
5. Écrire une fonction **`passage(L,s)`** qui prend en argument une liste **`L`** et une valeur à comparer **`s`** et qui renvoie tous les indices **`k`** des valeurs de **`L`** telles que **`L[k]<s<= L[k+1]`**.
6. Écrire une fonction **`monotonie(L)`** qui prend en argument une liste **`L`** et qui renvoie la liste des indices i_k tels que pour les indices entre i_k et i_{k+1}, la liste est monotone : les valeurs ne fonc que croître ou décroître de **`L[i[k]]`** à **`L[i[k+1]]`**.