DS 1 : lundi 14 octobre

2h sans calculatrice

Le candidat encadrera ou soulignera les résultats.
N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Exercice 1 *SQL (E3A MP 2018)*.
Nous nous intéressons à une base de données des zoos qui contient deux tables. La première table, `zoos`, a quatre colonnes dont `id`, un identifiant unique pour chaque zoo. Quelques lignes sont données ci-après.

id	nom	pays	continent
FR42	Zoo de La Flèche	France	Europe
RU12	Parc zoologique de Novossibirsk	Russie	Asie
RU5	Parc zoologique de Saint-Pétersbourg	Russie	Europe
⋮	⋮	⋮	⋮

La seconde table,`animaux`, a 6 colonnes, notamment un identifiant unique pour chaque animal (`id`) et l'identifiant du zoo qui héberge l'animal (`zoo`).

id	nom	espece	sexe	naissance	zoo
ke860	Kaiko	Chameau	F	2013	FR42
ic431	Jeffrey	Python royal	M	2016	RU12
gz599	Antaeus	Annaconda vert	M	2016	RU12
⋮	⋮	⋮	⋮	⋮	⋮

Les questions suivantes demandent d'écrire des requêtes SQL. À chaque fois quelques lignes de la table attendue sont données en exemple.

1° Donner les clefs primaires et étrangères des deux tables.

2° Écrire une requête SQL renvoyant la table des animaux (id, nom, naissance et zoo) nés avant 2010.

3° Écrire une requête SQL renvoyant la table des chamelles (chameaux femelles)classée par ordre alphabétique sur le nom.

id	nom	naissance	zoo
md375	Aimy	2012	CG01
ke860	Kaiko	2013	FR42
⋮	⋮	⋮	⋮

4° Écrire un requête SQL renvoyant la liste des animaux avec le nom du zoo et le pays où ils se trouvent.

5° Écrire une requête SQL renvoyant la table des bonobos mâles vivant en Asie.

id	nom	naissance	zoo
yv919	Finn	2008	CN33
qv139	Proteus	2013	KR08
⋮	⋮	⋮	⋮

6° Écrire une requête renvoyant la liste du nombre de zoo dans chaque pays.

7° Écrire une requête renvoyant la liste des pays ayant des zoos sur plusieurs continents.

pays
Russie
⋮

Exercice 2 *Code correcteur (Banque PT 2015)*.

Un signal transmis peut être perturbé par toutes sortes de facteurs pouvant provoquer des erreurs dans les données. En pratique, il est donc indispensable de pouvoir détecter ces erreurs et, dans la mesure du possible, les corriger sans que cela ne nécessite une nouvelle transmission ; l'objet de cette partie est de mettre en place quelques algorithmes dans ce but.

1° **Bit de parité.**

Une technique simple et très répandue pour s'assurer qu'une donnée sera lue correctement par son récepteur est de lui adjoindre un **bit de parité**, égal par définition à :

— 0 si la donnée contient un nombre pair de 1 (et, donc, si ses bits sont de somme paire),

— 1 si la donnée contient un nombre impair de 1 (et, donc, si ses bits sont de somme impaire).

Après réception de la donnée, le récepteur recalcule le bit de parité et le compare à celui que l'émetteur lui a adressé. Si la donnée n'a pas été altérée lors de la transmission, alors les deux bits de parité sont forcément identiques.

Q1. Donner les bits de parité associés aux représentations binaires des entiers 5, 16 et 37.

Q2. Écrire une fonction `parite(bits)` prenant pour argument une liste `bits` constituée d'entiers valant 0 ou 1 et retournant l'entier 0 ou 1 correspondant à son bit de parité.

Les techniques de vérification les plus simples consistent à découper la donnée en blocs et à joindre un bit de parité à chaque bloc. Par exemple, certains protocoles transmettent sept bits de données pour un bit de parité.

Q3. Donner un exemple d'erreur n'étant pas détectable par cette technique. Si une erreur a été détectée, est-il possible de la corriger sans retransmettre la donnée ?

2° Code de Hamming.

Le code de Hamming est un exemple d'utilisation des bits de parité pour détecter et corriger des erreurs. Nous nous intéressons ici au code dit (7,4), ainsi appelé car il consiste à joindre trois bits de parité à quatre bits de données, ce qui donne un message d'une longueur totale de sept bits. Ces trois bits de parité sont définis ainsi : si la donnée s'écrit (d_1, d_2, d_3, d_4) avec $d_i = 0$ ou 1, alors :
— p_1 est le bit de parité du triplet (d_1, d_2, d_4),
— p_2 est le bit de parité du triplet (d_1, d_3, d_4),
— p_3 est le bit de parité du triplet (d_2, d_3, d_4).
Le message encodé, que l'on transmet, s'écrit alors comme suit : $(p_1, p_2, d_1, p_3, d_2, d_3, d_4)$.

Q4. Écrire une fonction encode `encode_hamming(donnee)` prenant pour argument une liste `donnee` de quatre bits (représentés par des entiers valant 0 ou 1) et retournant une liste de bits contenant le message encodé.
On pourra appeler la fonction `parite(bits)` précédemment définie.

Le contrôle après réception d'un message ainsi encodé est relativement simple. On pourrait naturellement recalculer les trois bits de parité de la donnée et les comparer aux valeurs transmises, mais la technique proposée par Hamming est de calculer les trois bits de contrôle suivants, notés (c_1, c_2, c_3), à partir du message complet (données et bits supplémentaires), noté $(m_1, \ldots, m_7)$:
— c_1 est le bit de parité de l'ensemble (m_4, m_5, m_6, m_7),
— c_2 est le bit de parité de l'ensemble (m_2, m_3, m_6, m_7),
— c_2 est le bit de parité de l'ensemble (m_1, m_3, m_5, m_7).

On montre que si le message a bien été encodé selon les règles précédentes et n'a pas été altéré, alors les trois bits de contrôle doivent être à 0. Si ce n'est pas le cas, alors il y a eu une erreur ; l'intérêt de la technique de Hamming est que dans le cas particulier où l'erreur est unique, le mot de contrôle donne la représentation binaire de la position de cette erreur en numérotant à partir de 1. Par exemple, si $(c_1, c_2, c_3) = (0, 1, 1)$, alors l'erreur porte sur le troisième bit du message. Il suffit ainsi d'inverser ce bit (le mettre à 1 s'il est à 0, et inversement) pour corriger l'erreur.

La donnée décodée est alors constituée des quatre bits (d_1, d_2, d_3, d_4) qui se trouvent respectivement en positions 3, 5, 6 et 7 (toujours en numérotant à partir de 1) conformément à la description de l'encodage donnée ci-dessus.

Q5. Écrire une fonction `decode_hamming(message)` prenant pour argument une liste de sept bits et retournant une liste de quatre bits contenant la donnée décodée. En cas d'erreur, on affichera à l'écran un avertissement indiquant la position du bit affecté et on effectuera la correction. On supposera dans cette question que s'il y a une erreur, alors elle est unique.

Q6. Déterminer le codage de Hamming de la donnée 1011, puis la donnée décodée par l'algorithme dans l'hypothèse où les deux premiers bits du message codé ont été incorrectement transmis. Que à été l'effet de la "correction" sur la donnée dans ce cas ?

Q7. Sans coder, proposer un moyen simple de différencier une double erreur d'une erreur unique au moyen d'un bit de parité supplémentaire et expliquer comment cela permet d'éviter le problème mis en évidence à la question précédente. On s'appuiera sur les techniques introduites dans cette partie. On ne demande pas d'essayer de corriger la double erreur.

Exercice 3.

On considère l'algorithme suivant, qui s'applique à deux entiers naturels `i` et `j` préalablement définis. La fonction `floor` est la fonction partie entière (aussi bien en python que dans ce pseudo-code)

- `t` reçoit la valeur `0`.
- **Tant que** `i`$\geq$`1`, **faire** :
  - **Si** `i` est impair **alors** `t` reçoit la valeur `t+j` **Fin du si**.
  - `j` est multiplié par `2`.
  - `i` reçoit la valeur `floor(i/2)`.
  - **Fin du tant que**.
- `Renvoyer t`.

1. Traduire (on dit aussi implémenter) cet algorithme en Python sous la forme d'une fonction `mystere(i,j)` qui retournera la valeur finale de `t`.

2. Que se passe-t-il si `i` vaut 8 et `j` vaut 9 ? Et si `i` vaut 9 et `j` vaut 8 (on détaillera les étapes dans au moins l'un des deux cas) ?

3. Justifier que la boucle s'arrête.

4. Dans le cas général que fait cet algorithme ? Justifier que ce programme est correct avec un invariant de boucle.
   Indication : On pourra noter i_n, j_n et t_n les valeurs de i, j et t à la fin du n-ième passage dans la boucle et se demander comment évolue la valeur $i_n j_n + t_n$

Exercice 4.
Écrivez une fonction prenant une liste en entrée et renvoyant le booléen *True* si la liste est croissante (*False* sinon).