Math93.com

BAC NSI - Correction Amérique Nord - 2024 - Sujet 2 - NSI

Thèmes des exercices (sur 20 points)

- Exercice 1 (6 points) : Les algorithmes de tri et la programmation en Python.
- Exercice 2 (6 points) : Le langage SQL et les bases de données.
- Exercice 3 (8 points) : La programmation objet, les structures de données, les réseaux et l'architecture matérielle.
- Lien vers le sujet

Exercice 1. Les algorithmes de tri et la programmation en Python — 6 points

Cet exercice porte sur : Les algorithmes de tri et la programmation en Python.

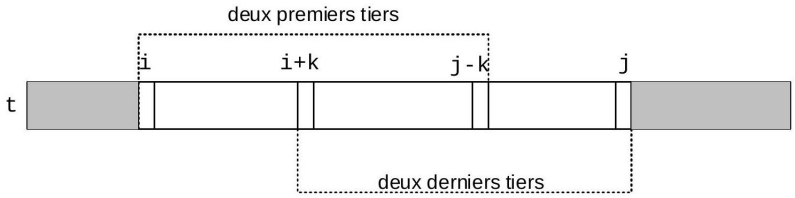
On se propose dans cet exercice de se pencher sur un algorithme pour trier un tableau appelé le tri de *Stooge*. Pour trier les éléments situés entre les indices i et j, où i < j, dans un tableau t par ce tri, on procède ainsi :

- si les éléments d'indice i et j sont mal placés, on les échange ;
- si il y a au moins trois éléments entre les indices i et j
  - on trie les deux premiers tiers du tableau avec cette méthode ;
  - on trie les deux derniers tiers du tableau avec cette méthode ;
  - on trie à nouveau les deux premiers tiers du tableau avec cette méthode.

Pour réaliser ce découpage en tiers, on considère l'entier k défini par l'expression

$$(j-i+1)//3$$

et on considère les indices intermédiaires $i+k$ et $j-k$.



Voici le code partiel de l'algorithme du tri de *Stooge* en Python qui trie donc les éléments d'un tableau par ordre croissant.

```
def triStooge(tab, i, j):
    if tab[i] > tab[j]:
        echange(tab, i, j)
    if (j - i) > 1:
        k = (j - i + 1)//3
        triStooge(...)
        triStooge(...)
        triStooge(...)
```

1. Écrire la fonction `echange(tab, i, j)` qui prend en arguments une liste Python `tab` et deux indices i, j, et réalise sur place l'échange des valeurs dans `tab` à ces indices. La fonction ne renvoie rien.

Corrigé

```
1  def echange(tab, i, j):
2      """Echange les valeurs dans tab aux indices i et j."""
3      temp = tab[i]
4      tab[i] = tab[j]
5      tab[j] = temp
6
7  # ou
8
9  def echange2(tab, i, j):
10     """Echange les valeurs dans tab aux indices i et j."""
11     tab[i],tab[j] = tab[j],tab[i]
```

2. Finaliser le programme précédent en complétant les lignes 6,7 et 8 sur votre copie.

Corrigé

```
1  def triStooge(tab, i, j):
2      if tab[i] > tab[j]:
3          echange(tab, i, j)
4      if (j - i) > 1:
5          k = (j - i + 1)//3
6          triStooge(tab , i , j-k) # on trie les deux premiers tiers du tableau
7          triStooge(tab , i+k , j) # on trie les deux derniers tiers du tableau
8          triStooge(tab , i , j-k) # on trie à nouveau les deux 1er tiers du ta
```

3. Indiquer en le justifiant si cet algorithme est itératif ou récursif.

Corrigé

L'algorithme `triStooge` est récursif car il s'appelle lui-même à l'intérieur de sa propre définition pour résoudre des sous-problèmes plus petits.

Soit l'appel `triStooge(A, 0, 5)` avec $A = [5,6,4,2,3,1]$.

4. Déterminer la valeur numérique prise par k lors de ce premier appel. Une justification est attendue.

Corrigé

La valeur numérique prise par k lors de ce premier appel est :

$$\begin{aligned} k &= (j-i+1)//3 \\ &= (5-0+1)//3 \\ k &= 6//3 = 2 \end{aligned}$$

La figure ci-dessous présente l'arbre (incomplet) des appels récursifs effectués depuis l'appel `triStooge(A, 0, 5)`.

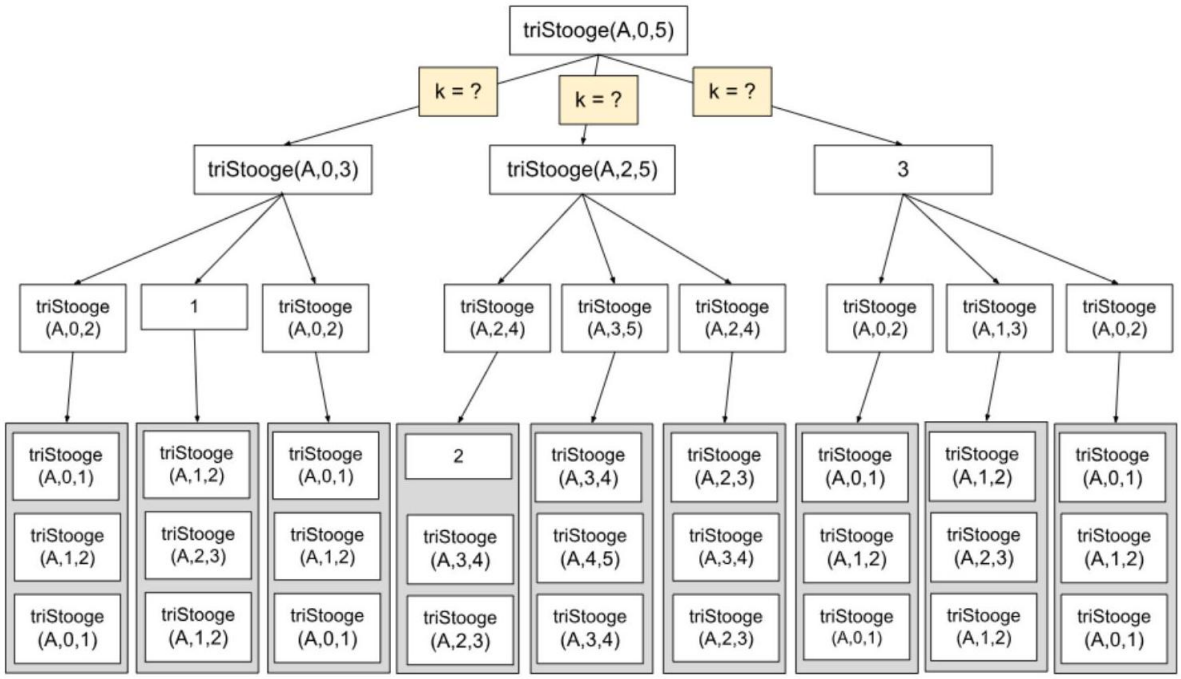


Figure 1. Arbre des appels récursifs pour `triStooge(A, 0, 5)`

5. Dénombrer le nombre d'appels récursifs effectués lors du tri sans compter l'appel initial.

Corrigé

On compte les cases de la figure 1, on en trouve 39.

6. Déterminer les appels effectués dans les cases 1, 2 et 3 de cet arbre des appels récursifs.

Corrigé

- case 1 : `triStooge(A, 1, 3)`
- case 2 : `triStooge(A, 2, 3)`
- case 3 : `triStooge(A, 0, 3)`

7. Dans cette question, A = [5,4,6,2]. Recopier le tableau ci-dessous et le compléter (remplacer les ??).

Appel	Valeur de A avant l'appel	Valeur de A après l'appel
triStooge(A, 0,3)	[5,6,4,2]	??
??	[2,6,4,5]	??
triStooge(A, 1, 3)	[2,4,6,5]	??
??	[2,4,5,6]	[2,4,5,6]

Corrigé

Erreur dans cette question car le tableau ne correspond pas à la question.

On montre que le coût en temps dans le pire des cas de l'algorithme de Stooge est de l'ordre de n^e, avec e environ égal à $\frac{8}{3}$.

8. Donner un algorithme de tri dont le coût est strictement meilleur.

Corrigé

L'algorithme de tri-fusion a un coût meilleur en $O\left(n.\log_2(n)\right)$.

De même le tri par sélection ou par insertion en $O\left(n^2\right)$ car $2 < \frac{8}{3}$

Exercice 2. Le langage SQL et les bases de données — 6 points

Cet exercice porte sur : Le langage SQL et les bases de données.

Un pharmacien nouvellement installé décide de créer son propre système de gestion des médicaments qu'il délivre à ses clients.
Pour sa base de données relationnelle, il a déjà élaboré la première relation à l'aide des données indiquées sur les cartes vitales de ses deux premiers clients :

```
client (id_client : INT, nom_client : VARCHAR(30),
        prenom_client : VARCHAR(30), num_secu_sociale : VARCHAR(15))
```

client			
id_client	nom_client	prenom_client	num_secu_sociale
1	Martin	Sophie	202103812326129
2	Dufour	Marc	105073817009595

1. Écrire le résultat de l'exécution de la requête SQL suivante :

```
SELECT nom_client, prenom_client
FROM client
ORDER BY nom_client;
```

Corrigé

nom_client	prenom_client
Dufour	Marc
Martin	Sophie

Pour écrire la relation medicament, il doit utiliser les informations fournies par la notice des médicaments. En voici une ci-dessous :

Paracétamol 1 Cramme CP
Qu'est-ce que Paracétamol 1 gramme CP et dans quel cas est-il utilisé ?
Paracétamol 1 gramme CP est un antalgique (calme la douleur)
Que contient un comprimé de Paracétamol 1 gramme CP ?
La substance active est le paracétamol : 1 gramme pour un comprimé.
Sous quelle forme se présente Paracétamol 1 gramme CP ?
Ce médicament se présente sous la forme de comprimé.
Chaque boite contient 8 comprimés.

Figure 1. Informations extraites de la notice du médicament Paracétamol 1 gramme CP.

La relation `medicament` suivante a été obtenue à l'aide de ces notices :

```
medicament (id_medic : INT, nom_medic : VARCHAR(30),
            categorie : VARCHAR(20), conditionnement : INT,
           quantite : INT, prix : FLOAT)
```

La table des médicaments de son officine est présentée ci-dessous.

medicament					
id_medic	nom_medic	categorie	conditionnement	quantite	prix
1	Paracétamol 1 gramme CP	antalgique	8	50	3,50
2	Acide acétylsalicylique	antalgique	8	20	2,30
3	Gel hydroalcoolique 100 ml	désinfectant	1	300	2,30
4	Acide ascorbique	vitamine	10	450	5,50

2. Écrire une requête *SQL* permettant d'afficher les noms de tous les médicaments dont le prix est strictement inférieur à 3 euros.

. Corrigé

```
SELECT nom_medic
FROM medicament
WHERE prix < 3;
```

Madame Martin présente au pharmacien une nouvelle ordonnance :

CENTRE MEDICAL

Dr Louis FARTI
Grenoble, le 13 décembre 2023
Mme Sophie MARTIN

- Paracétamol 1 gramme CP (boite de 8)
  Prendre, par voie orale, 1 comprimé par prise, à renouveler en cas de besoin au bout de 4 heures minimum, avec au maximum 3 comprimés par jour, pendant 2 jours.
- Acide ascorbique (vitamine C) 500mg comprimé effervescent (boite de 10)
  Un comprimé par jour pendant 4 semaines.

Figure 2. Ordonnance de Madame Sophie Martin.

Il saisit les informations de cette ordonnance dans la relation ordonnance, chaque médicament prescrit correspondant à un enregistrement dans la table ci-dessous.

ordonnance				
id_ordo	id_client	date_ordo	id_medic	nb_boites
6	2	2023 – 11 – 29	2	2
7	1	2023 – 12 – 13	1	…
8	1	2023 – 12 – 13	4	…

3. Écrire une requête *SQL* permettant d'ajouter les informations de la carte vitale de sa troisième cliente présentée ci-dessous :

Figure 3. Image de la carte vitale extraite de la page wikipédia

Source : d'après https ://fr.wikipedia.org/wiki/Carte_Vitale (wikipedia.org)

Corrigé

```
INSERT INTO client (id_client, nom_client, prenom_client, num_secu_sociale)
VALUES (3, 'Durand', 'Claire', '306047812345678');
```

4. Donner les attributs qui doivent être déclarés comme clés étrangères de la relation ordonnance et en préciser l'utilité.

Corrigé

Les attributs qui doivent être déclarés comme clés étrangères dans la relation `ordonnance` sont :

- `id_client` : clé étrangère vers la table `client`. Cela permet d'associer une ordonnance à un client spécifique.
- `id_medic` : clé étrangère vers la table `medicament`. Cela permet d'associer une ordonnance à un médicament spécifique.

L'utilité des clés étrangères est de maintenir l'intégrité référentielle entre les tables, en s'assurant que les enregistrements dans la table `ordonnance` correspondent à des enregistrements valides dans les tables `client` et `medicament`.

5. Indiquer, pour les lignes 7 et 8 de la table ordonnance, le nombre de boites prescrites.

Corrigé

Pour la ligne 7 de la table `ordonnance` :

- `id_ordo` : 7
- `id_client` : 1
- `date_ordo` : 2023-12-13
- `id_medic` : 1 (Paracétamol 1 gramme CP)
- `nb_boites` : 1 boîte (max 3 comprimés par jour pendant 2 jours soit 6, donc une boîte de 8 comprimés est suffisante)

Pour la ligne 8 de la table `ordonnance` :

- `id_ordo` : 8
- `id_client` : 1
- `date_ordo` : 2023-12-13
- `id_medic` : 4 (Acide ascorbique)
- `nb_boites` : 3 boîtes (1 comprimé par jour pendant 4 semaines, donc 3 boîtes de 10 comprimés)

6. Écrire la requête SQL mettant à jour la quantité du médicament Acide ascorbique en stock dans l'officine du pharmacien suite au passage de Madame Martin.

Corrigé

```sql
UPDATE medicament
SET quantite = quantite - 3
WHERE id_medic = 4;
```

7. Calculer le coût total des médicaments fournis à Madame Martin (on ne demande pas d'écrire une requête ici, mais de calculer le coût total en justifiant le calcul).

Corrigé

- Pour le Paracétamol 1 gramme CP : 1 boîte à 3,50 euros.
- Pour l'Acide ascorbique : 3 boîtes à 5,50 euros chacune.

Coût total :

$$1 \times 3{,}50 + 3 \times 5{,}50 = 3{,}50 + 16{,}50 = 20€$$

8. Écrire la requête SQL permettant d'afficher le nom du médicament pour l'ordonnance ayant l'id_ordo numéro 6.

. Corrigé

```sql
SELECT nom_medic
FROM medicament
WHERE id_medic = (
  SELECT id_medic
  FROM ordonnance
  WHERE id_ordo = 6
);

OU

SELECT nom_medic
FROM medicament
JOIN ordonnance ON ordonnance.id_medic = medicament.id_medic
WHERE id_ordo = 6;
```

Exercice 3. La programmation objet, les structures de données, les réseaux et l'architecture matérielle — 8 points

Cet exercice porte sur : La programmation objet, les structures de données, les réseaux et l'architecture matérielle.

On considère un réseau local constitué des trois machines de Alice, Bob et Charlie dont les adresses IP sont les suivantes :

- la machine d'Alice a pour adresse `192.168.1.1` ;
- la machine de Bob a pour adresse `192.168.1.2` .

On rappelle que l'adresse `192.168.1.255` est l'adresse de diffusion qui sert à communiquer avec toutes les machines du réseau local et le masque de ce réseau local est `255.255.255.0` . Cette adresse de diffusion est réservée et ne peut être attribuée à une machine.

Partie A

1. Donner une adresse IP possible pour la machine de Charlie afin qu'elle puisse communiquer avec celles d'Alice et Bob dans le réseau local. Justifier votre réponse en donnant toutes les conditions à respecter dans le choix de cette adresse IP.

Corrigé

Une adresse IP possible pour la machine de Charlie pourrait être `192.168.1.3`. Les conditions à respecter sont :

- L'adresse IP doit être dans le même sous-réseau que les machines d'Alice et Bob, ce qui est assuré par le masque `255.255.255.0`. Les adresses valides vont de `192.168.1.1` à `192.168.1.254` car l'adresse de diffusion `192.168.1.255` et l'adresse du réseau `192.168.1.0` sont réservées.
- L'adresse IP doit être unique au sein du réseau local donc on exclut les adresses de Bob `192.168.1.2` et Alice `192.168.1.1` .

Ce réseau est utilisé pour effectuer des transactions financières en monnaie numérique `nsicoin` entre les trois utilisateurs. Pour cela, on crée la classe `Transaction` ci-dessous :

```
class Transaction:
    def __ini__(self, expediteur, destinataire, montant):
        self.expediteur = expediteur
        self.destinataire = destinataire
        self.montant = montant
```

2. Toutes les dix minutes, les transactions réalisées pendant cet intervalle de temps sont regroupées par ordre d'apparition dans une liste Python. Dans un intervalle de dix minutes, Alice envoie dix `nsicoin` à Charlie puis Bob envoie cinq `nsicoin` à Alice. Écrire la liste Python correspondante à ces transactions.

. Corrigé

```
t1 = Transaction("Alice", "Charlie", 10)
t2 = Transaction("Bob", "Alice", 5)

transaction=[t1,t2]

# ou directement

transaction2=
[
Transaction("Alice", "Charlie", 10),
Transaction("Bob", "Alice", 5)
]
```

Pour garder une trace de toutes les transactions effectuées, on utilise une liste chaînée de blocs (ou *blockchain*) dont le code Python est fourni ci-dessous. Toutes les dix minutes un nouveau bloc contenant les nouvelles transactions est créé et ajouté à la *blockchain*.

```
class Bloc:
    def __init__(self, liste_transactions, bloc_precedent):
        self.liste_transactions = liste_transactions
        self.bloc_precedent = bloc_precedent # de type Bloc

class Blockchain:
    def __init__(self):
        self.tete = self.creer_bloc_0()

    def self.creer_bloc_0(self):
         """
         Crée le premier bloc qui distribue 100  nsicoin  à tous les utilisateurs
         (un pseudo-utilisateur Genesis est utilisé comme expéditeur)
         """
        liste_transactions = [
            Transaction("Genesis", "Alice", 100),
            Transaction("Genesis", "Bob", 100),
            Transaction("Genesis", "Charlie", 100)
            ]
        return Bloc(liste_transactions, None)

# ERREUR , on devrait ligne 10 écrire
def creer_bloc_0(self):
```

3. La figure 1 représente les trois premiers blocs d'une Blockchain. Expliquer pourquoi la valeur de l'attribut `bloc_precedent` du bloc0 est `None` .

Corrigé

La valeur de l'attribut `bloc_precedent` du bloc0 est `None` car le bloc0 est le premier bloc de la blockchain, également appelé bloc Genesis. Il n'y a aucun bloc précédent auquel il pourrait se lier.

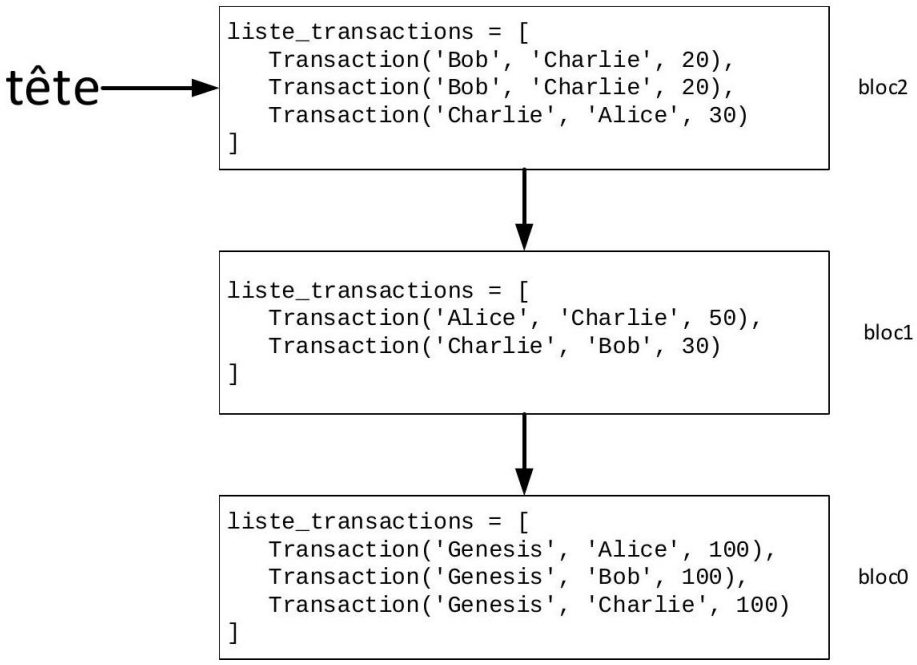


Figure 1. Blockchain

4. Donner la valeur de l'attribut `bloc_precedent` du bloc1 afin que celui-ci soit lié au bloc0.

Corrigé

La valeur de l'attribut `bloc_precedent` du bloc1 doit être `bloc0`.

5. À l'aide des classes `Bloc` et `Blockchain` , écrire le code Python permettant de créer un objet `ma_blockchain` de type `Blockchain` représenté par la figure 1.

Corrigé

```
ma_blockchain = Blockchain()

bloc1 = Bloc(
[Transaction('Alice', 'Charlie', 50),
Transaction('Charlie','Bob', 30)],
ma_blockchain.tete)

ma_blockchain.tete = bloc1

bloc2 = Bloc(
[Transaction('Bob', 'Charlie', 20),
Transaction('Bob','Charlie', 20),
Transaction('Charlie', 'Alice', 30)],
ma_blockchain.tete)

ma_blockchain.tete = bloc2
```

6. Donner le solde en `nsicoin` de Bob à l'issue du bloc2.

Corrigé

Le solde de Bob est :

$$100 + 30 - 20 - 20 = 90$$

7. On souhaite doter la classe `Blockchain` d'une méthode `ajouter_bloc` qui prend en paramètre la liste des transactions des dix dernières minutes et l'ajoute dans un nouveau bloc. Écrire le code Python de cette méthode cidessous.

```
def ajouter_bloc(self, liste_transactions):
    # A compléter
```

Corrigé

```
def ajouter_bloc(self, liste_transactions):
    nouveau_bloc = Bloc(liste_transactions, self.tete)
    self.tete = nouveau_bloc
```

8. Lorsqu'un utilisateur ajoute un nouveau bloc à la Blockchain, il l'envoie aux autres membres. Ainsi chaque utilisateur dispose sur sa propre machine d'une copie identique de la Blockchain. Donner le nom et la valeur de l'adresse IP à utiliser pour effectuer cet envoi.

Corrigé

Il faut utiliser ladresse de broadcast (adresse de diffusion) : `192.168.1.255`

9. On souhaite doter la classe `Bloc` d'une nouvelle méthode `calculer_solde` permettant de renvoyer le solde à l'issue de ce bloc. Recopier et compléter sur votre copie le code Python de cette méthode :

```
def calculer_solde(self, utilisateur):
    if self.precedent is None: # cas de base
        solde = 0
    else:
        solde = . . . # appel récursif : calcul du solde au bloc précédent
    for transaction in bloc.liste_transactions
        if ... == utilisateur:
            solde = solde - ....
        elif ... :
            ...
    return solde
```

Corrigé

```python
def calculer_solde(self, utilisateur):
    if self.precedent is None: # cas de base
        solde = 0
    else:
        solde = self.bloc_precedent.calculer_solde(utilisateur) # appel récursif
    for transaction in bloc.liste_transactions
        if transaction.expediteur  == utilisateur:
            solde = solde - transaction.montant
        elif transaction.destinataire == utilisateur :
            solde = solde + transaction.montant
    return solde


```

10. Écrire l'appel à la fonction `calculer_solde` permettant de calculer le solde actuel de Alice

. Corrigé

```python
ma_blockchain.calculer_solde('Alice')
```

Partie B

Dans cette partie, on va améliorer la sécurité de la blockchain. Pour cela on enrichit la classe `Bloc` comme indiqué ci-dessous :

```python
class Bloc:
    def __init__(self, liste_transactions, bloc_precedent):
    self.liste_transactions = liste_transactions
    self.bloc_precedent = bloc_precedent
    # Définition de trois nouveaux attributs
    self.hash_bloc_precedent = self.donner_hash_precedent()
    self.nonce = 0 # Fixé arbitrairement et temporairement à 0 avant le minage du
    self.hash = self.calculer_hash()

    # Définition de trois nouvelles méthodes
    def donner_hash_precedent(self):
        if self.bloc_precedent is not None:
            return self.bloc_precedent.hash
        else :
            return "0"

    def calculer_hash(self):
        """calcule et rencoie le hash du bloc courant"""
        # Le code python n'est pas étudié dans cet exercice

    def minage_bloc(self):
        """calcule le nonce d'un bloc pour que son hash commence par '00'"""
        # A compléter
```

La fonction `calculer_hash` produit une chaîne de caractères appelée hash qui possède les propriétés suivantes :

- Le hash d'un bloc dépend de toutes les données contenues dans le bloc et uniquement de ces données ;
- Le calcul du hash d'un bloc est rapide et facile à calculer par une machine ;
- La moindre modification dans le bloc produit un hash complètement différent ;
- II est impossible de déduire le bloc à partir de son hash.
- Si deux blocs ont le même hash, c'est qu'ils sont parfaitement identiques.

11. L'attribut `nonce` est de type entier. Miner un bloc signifie trouver une valeur de `nonce` de telle façon que l'attribut `hash` du bloc commence par les deux caractères " 00 ". Compte tenu des propriétés précédentes, la seule façon de trouver cette valeur est de procéder à une recherche exhaustive. Expliquer en quoi consiste le fait de trouver une valeur par recherche exhaustive.

Corrigé

Une recherche exhaustive consiste à essayer toutes les valeurs de `nonce` jusqu'à trouver un résultat qui convient.

Dans la suite de l'exercice, on considère que tous les utilisateurs cherchent à miner le nouveau bloc. Le premier qui réussit, l'ajoute à la blockchain et gagne une récompense en `nsicoin` .

12. En justifiant votre réponse, donner la valeur de l'attribut `hash_bloc_precedent` du bloc0.

Corrigé

Comme l'attribut du `_bloc_precedent` du bloc0 est `None` `hash_bloc_precedent="0"`

13. Sachant que le `hash` est écrit sur 256 bits, donner le calcul permettant d'obtenir le nombre de hash possibles.

Corrigé

Sur 256 bits il y a 2^{256} hashs possibles.

14. Recopier et compléter sur votre copie le code python de la méthode `minage_bloc` .

```
def minage_bloc(self):
    """modifie le nonce d'un bloc pour que son hash commence par '00'
    en énumérant tous les entiers naturels en partant de 0."""
    self.nonce = 0
    self.hash = self.calculer_hash()
    while ... :
        self.nonce = ...
        self.hash = ...
```

Corrigé

```
def minage_bloc(self):
    """modifie le nonce d'un bloc pour que son hash commence par '00'
    en énumérant tous les entiers naturels en partant de 0."""
    self.nonce = 0
    self.hash = self.calculer_hash()
    while self.hash [:2]!="00" :
        self.nonce =self.nonce+1
        self.hash = self.calculer_hash()


```