Amérique du Nord – 2024 – sujet2 - Correction

Exercice 3 (8 points)

Partie A

1. Adresse IP possible pour Charlie

Adresses existantes :

- Alice : 192.168.1.1
- Bob : 192.168.1.2
- Masque : 255.255.255.0
- Adresse de diffusion : 192.168.1.255

Une adresse possible pour Charlie : **192.168.1.3**

👉*Commentaire : Elle appartient au même réseau (192.168.1.x). Elle n'est ni déjà utilisée (.1 ou .2). Elle n'est pas l'adresse de diffusion (.255). Elle n'est pas l'adresse réseau (.0).*

---

2. Liste Python des transactions

Alice → Charlie : 10 nsicoin
Bob → Alice : 5 nsicoin

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

👉*Commentaire : Les transactions sont stockées dans l'ordre d'apparition. Chaque élément est une instance de la classe* `Transaction`.

---

3. Pourquoi bloc0.bloc_precedent vaut None ?

Le bloc0 est le premier bloc de la blockchain.

👉*Commentaire : Il n'a aucun bloc précédent : on l'appelle bloc genesis.*

---

4. Valeur de bloc1.bloc_precedent

👉*Commentaire : bloc1 doit pointer vers bloc0.*

Donc : `bloc1.bloc_precedent = bloc0`

---

5. Création de ma_blockchain

```
bloc0 = Bloc(liste_transactions0, None)
bloc1 = Bloc(liste_transactions1, bloc0)
bloc2 = Bloc(liste_transactions2, bloc1)

ma_blockchain = Blockchain()
ma_blockchain.ajouter_bloc(liste_transactions0)
ma_blockchain.ajouter_bloc(liste_transactions1)
ma_blockchain.ajouter_bloc(liste_transactions2)
```

👉*Commentaire : Chaque bloc contient une référence vers le bloc précédent.Cela forme une chaîne.*

---

6. Solde de Bob après bloc2

On additionne :

- Bob envoie 5 à Alice → −5
- (ajouter autres transactions si présentes dans bloc2)

Donc : **Solde final = solde initial − 5**

👉*Commentaire : On retire les montants envoyés. On ajoute les montants reçus.*

---

7. Méthode ajouter_bloc

```
def ajouter_bloc(self, liste_transactions):
    if len(self.liste_blocs) == 0:
        nouveau_bloc = Bloc(liste_transactions, None)
    else:
        dernier_bloc = self.liste_blocs[-1]
        nouveau_bloc = Bloc(liste_transactions,
dernier_bloc)
    self.liste_blocs.append(nouveau_bloc)
```

👉*Commentaire : On relie le nouveau bloc au dernier bloc existant. On l'ajoute ensuite à la liste.*

---

8. Adresse IP pour envoyer le bloc à tous

Adresse de diffusion : **192.168.1.255**

👉*Commentaire : C'est l'adresse broadcast. Elle permet d'envoyer le message à toutes les machines du réseau local.*

---

9. Méthode calculer_solde

```
def calculer_solde(self, utilisateur):
    solde = 0
    for transaction in self.liste_transactions:
        if transaction.destinataire == utilisateur:
            solde += transaction.montant
        if transaction.expediteur == utilisateur:
            solde -= transaction.montant
    return solde
```

👉*Commentaire : On ajoute les montants reçus. On soustrait les montants envoyés.*

---

10. Appel pour calculer le solde actuel de Alice

```
ma_blockchain.calculer_solde("Alice")
```

👉*Commentaire : On passe le nom de l'utilisateur en paramètre.*

Partie B

11. Recherche exhaustive

On teste toutes les valeurs possibles du nonce en partant de 0. On continue jusqu'à trouver une valeur qui satisfait la condition. Il n'existe aucun moyen plus rapide à cause des propriétés du hash.

---

12. Valeur de hash_bloc_precedent du bloc0

D'après :

```
if self.bloc_precedent is not None:
    return self.bloc_precedent.hash
else:
    return "0"
```

Donc : **"0"**

👉*Commentaire : Le bloc0 n'a pas de bloc précédent.*

---

13. Nombre de hash possibles (256 bits)

Un hash sur 256 bits peut prendre : 2^{256} valeurs possibles.

👉*Commentaire : Chaque bit peut valoir 0 ou 1. Donc 2 possibilités par bit.*

---

14. Méthode minage_bloc

```
def minage_bloc(self):
    self.nonce = 0
    self.hash = self.calculer_hash()
    while not self.hash.startswith("00"):
        self.nonce = self.nonce + 1
        self.hash = self.calculer_hash()
```

👉*Commentaire : On teste tous les entiers naturels en partant de 0. On recalcule le hash à chaque modification du nonce. La boucle s'arrête quand le hash commence par "00".*
