Examen Blockchain : Énoncés Complets et Correction

1. **Dans un système blockchain, les blocs sont organisés dans une séquence linéaire au fil du temps en forçant chaque entête de bloc à contenir :**
   A. Le hash du bloc suivant
   B. Le hash du bloc précédent
   C. Les hash de tous les blocs antérieurs
   D. Le hash de tous les futurs blocs
   Justification : C'est le principe fondamental du "chaînage". Si on modifie un bloc passé, son hash change, ce qui invalide tous les blocs suivants.
2. **Afin d'authentifier une transaction comme valide, un système blockchain exige que :**
   A. L'expéditeur signe digitalement la transaction
   B. Le receveur signe digitalement la transaction
   C. L'expéditeur sache quelle est la clé privée du récepteur
   D. Le receveur sache quelle est la clé privée de l'envoyeur
   Justification : La signature prouve que le détenteur de la clé privée (le propriétaire des fonds) a autorisé le transfert.
3. **Afin de signer et diffuser une transaction blockchain, les utilisateurs utiliseront d'habitude :**
   A. Le logiciel d'exploitation minière
   B. Un explorateur blockchain
   C. Un portefeuille (wallet)
   Justification : Le wallet est l'interface utilisateur qui gère les clés et communique avec le réseau.
4. **Dans un système blockchain, une adresse de compte est généralement calculée en tant que :**
   A. Formulaire sérialisé de la clé privée
   B. Le hash de la clé privée
   C. Le hash de la clé publique
   D. Le hash de toutes les transactions sortantes passées de ce compte
   Justification : Pour des raisons de sécurité et de formatage, on applique une fonction de hachage (comme SHA-256 ou Keccak-256) à la clé publique.
5. **Dans une blockchain preuve de travail (proof of work), les mineurs génèrent de nouveaux blocs valides en :**
   A. Hashant à plusieurs reprises le contenu du bloc précédent
   B. Validant à plusieurs reprises les signatures de transaction du bloc actuel
   C. Hashant à plusieurs reprises
   Justification : Le minage est une compétition de puissance de calcul consistant à essayer des millions de "nonces" pour obtenir un hash spécifique.
6. **Dans une blockchain publique, les contenus des blocs, y compris les données de transaction, peuvent être lus, explorés et vérifiés :**
   A. Uniquement par les titulaires de compte
   B. Uniquement par les validateurs ou les mineurs du réseau
   C. Uniquement par plus de 50% des validateurs ou mineurs du réseau
   D. Par chacun connecté au réseau
   Justification : La transparence totale est le pilier d'une blockchain publique (ex : Bitcoin, Ethereum).
7. **Comment ajoutez-vous un nouvel élément à la fin d'un tableau en Solidity ?**
   A. tableau1.push(4) ;
   B. tableau1.add(4) ;

C. tableau1.append(4) ;
D. tableau1.insert(4) ;
Justification : La fonction native en Solidity pour les tableaux dynamiques est `.push()`.

8. **Étant donné un message M et une fonction hash H, cocher la réponse valide.**
A. Si H(M) est connu, vous pouvez calculer M facilement.
B. Si M est connu, vous pouvez calculer H(M) facilement.
C. Il est garanti que H(M) et M ont toujours la même longueur.
Justification : Une fonction de hachage est une fonction à sens unique : facile à créer, presque impossible à inverser.

9. **Quelle est la fonction du réseau de test dans MetaMask ?**
A. Tester la vitesse de la connexion Internet
B. Expérimenter avec des transactions fictives sur une blockchain de test
C. Mettre en place un réseau social décentralisé
D. Vérifier l'authenticité des contrats intelligents
Justification : Les réseaux de test (testnets) permettent aux développeurs de tester leur code sans risquer de l'argent réel.

10. **Quelle formule représente le calcul du hash d'un bloc dans la preuve de travail ?**
A. Hash = Nonce * Données du bloc
B. Hash = Fonction_hashage(Données du bloc + Nonce)
C. Hash = Données du bloc / Nonce
D. Hash = Données du bloc - Nonce
Justification : On combine (concatène) les données fixes du bloc avec un nombre variable (le nonce) pour obtenir un hash différent à chaque essai.

11. **Quelle méthode Python permet d'ajouter un nouveau bloc à une chaîne de blocs ?**
A. add_block(new_block)
B. append(new_block)
C. insert(new_block)
D. update(new_block)
Justification : Dans les exercices de programmation de blockchain simple en Python, la chaîne est représentée par une liste (`list`). La méthode pour ajouter à une liste est `append()`.

12. **Quel est le rôle principal de l'instruction 'ASSERT' dans un contrat intelligent ?**
A. Vérifier une condition et interrompre l'exécution du contrat en cas d'échec.
B. Exécuter une condition sans affecter le flux du contrat.
C. Générer des erreurs lorsqu'une condition est vraie.
D. Valider l'authenticité du contrat.
Justification : `assert` est utilisé pour vérifier les erreurs qui ne devraient théoriquement jamais arriver si le code est correct.

13. **Que se passe-t-il si une instruction 'ASSERT' échoue dans un contrat intelligent ?**
A. La transaction est annulée et les changements sont annulés.
B. Une nouvelle tentative d'exécution de la transaction est effectuée.
C. Le contrat continue à s'exécuter normalement malgré l'échec.
D. Une notification est envoyée à l'administrateur du contrat.
Justification : C'est ce qu'on appelle un "revert". L'état de la blockchain revient exactement comme il était avant la transaction.

14. **Quel est le rôle de msg.sender dans un contrat intelligent Solidity ?**
A. Obtenir le solde du contrat
B. Identifier l'expéditeur de la transaction actuelle
C. Réaliser une opération mathématique
D. Appeler une fonction externe

Justification : C'est une variable globale qui permet de savoir quelle adresse a déclenché l'exécution de la fonction.

15. **Quel type de données est msg.sender en Solidity ?**
    A. String
    B. Bool
    C. Address
    D. Uint
    Justification : `msg.sender` représente une adresse Ethereum (20 octets), donc le type est `address`.

16. **Qu'est-ce que le "nonce" dans le contexte de la preuve de travail ?**
    E. Un nombre aléatoire ajusté pour trouver un hash valide
    F. Une clé privée d'un utilisateur
    G. Le nombre total de transactions dans un bloc
    H. Une unité de mesure du temps dans le minage
    Justification : C'est le seul paramètre que les mineurs ont le droit de modifier pour tenter d'obtenir un hash commençant par un certain nombre de zéros.

17. **Comment pouvez-vous utiliser msg.sender pour restreindre l'accès à une fonction uniquement à l'expéditeur d'une transaction ?**
    A. require(msg.sender == true) ;
    B. require(msg.sender == address(this)) ;
    C. require(msg.sender != address(0)) ;
    D. require(msg.sender == owner) ;
    Justification : En comparant l'appelant (`msg.sender`) à l'adresse du propriétaire stockée lors du déploiement (`owner`).

18. **Comment déclarez-vous une fonction qui accepte deux entiers en paramètres et renvoie leur somme en Solidity ?**
    A. function add(int a, int b) public pure returns (int) { return a + b ; }
    B. function add(int a, int b) public pure int { return a + b ; }
    C. function add(int a, int b) public pure { return a + b ; }
    D. function add(int a, int b) public pure returns { return a + b ; }
    Justification : La syntaxe correcte exige `returns (type)` avant le corps de la fonction.

19. **Comment déclarez-vous une fonction en Solidity qui prend un nombre entier en paramètre et renvoie le résultat de sa division par 3 ?**
    A. function divideByThree(int x) public pure returns (int) { return x / 3 ; }
    B. function divideByThree(int x) public pure { return x / 3 ; }
    C. function divideByThree(int x) public pure int { return x / 3 ; }
    D. function divideByThree(int x) public pure returns { return x / 3 ; }
    Justification : Même règle que précédemment : le mot-clé `returns` est indispensable pour renvoyer une valeur.

20. **Plusieurs réponses sont acceptées : (Signature électronique)**
    A. Le message est signé en utilisant la clé privée
    B. Le message est signé en utilisant la clé publique
    C. La signature est vérifiée en utilisant la clé privée
    D. La signature est vérifiée en utilisant la clé publique
    Justification : Dans la cryptographie asymétrique, on signe avec sa clé secrète (privée) pour prouver son identité, et tout le monde peut vérifier avec la clé publique correspondante.