QCM : Goroutines en Go

1. Que permet une goroutine en Go ?

- A. Appeler une fonction récursivement
- B. Exécuter une fonction de manière concurrente
- C. Optimiser l'utilisation de la mémoire
- D. Exécuter un programme en mode débogage

2. Quelle syntaxe permet de lancer une goroutine ?

- A. run maFonction()
- B. execute maFonction()
- C. go maFonction()
- D. goroutine maFonction()

3. Quel est le comportement d'une goroutine si le main() se termine avant elle ?

- A. Elle continue en arrière-plan
- B. Elle passe en mode veille
- C. Elle est arrêtée immédiatement
- D. Elle provoque une erreur

4. Le mot-clé go peut être utilisé avec :

- A. des variables uniquement
- B. des fonctions uniquement
- C. des méthodes ou fonctions
- D. n'importe quelle instruction

5. Quel paquet est utilisé pour synchroniser des goroutines ?

- A. fmt
- B. time
- C. sync
- D. goroutine

6. Quelle méthode du WaitGroup bloque jusqu'à la fin des goroutines ?

- A. Wait()
- B. Stop()

- C. Done()

7. Quelle est la meilleure façon d'assurer qu'un Mutex sera libéré ?

- A. En utilisant Unlock() tout de suite
- B. Avec une goroutine dédiée
- C. En appelant defer mu.Unlock()
- D. En supprimant le Lock()

8. Quelle situation illustre un *data race* ?

- A. Deux goroutines lisent la même variable
- B. Une goroutine lit, l'autre écrit la même variable sans synchronisation
- C. Deux goroutines font fmt.Println()
- D. Une goroutine dort pendant 1 seconde

9. Que fait sync.Mutex ?

- A. Interrompt une goroutine
- B. Laisse plusieurs goroutines accéder librement à une variable
- C. Synchronise des timers
- D. Empêche des goroutines d'accéder simultanément à une ressource partagée

10. Quel est le coût mémoire approximatif d'une goroutine ?

- A. 4 Go
- B. 1 Mo
- C. Quelques ko
- D. 500 Mo

11. Quelle fonction Go permet de faire une pause dans une goroutine ?

- A. wait()
- B. delay()
- C. sleep()
- D. time.Sleep()

12. Que signifie defer wg.Done() dans une goroutine ?

- A. Ne rien faire
- B. Bloquer la goroutine
- C. Signaler que la tâche est terminée à un WaitGroup

- D. Créer une nouvelle goroutine

13. Peut-on lancer une méthode anonyme en goroutine ?

- A. Non
- B. Oui
- C. Seulement avec main()
- D. Seulement si elle retourne un booléen

14. Quel est le risque si on oublie wg.Add(n) avant go ?

- A. Rien, Go le détecte
- B. wg.Wait() retourne directement
- C. Le programme panique
- D. Les goroutines ne s'exécutent pas

15. Peut-on utiliser un Mutex dans plusieurs fonctions ?

- A. Non, un mutex est local à une fonction
- B. Oui, s'il est passé en paramètre ou global
- C. Uniquement dans le main()
- D. Cela dépend du compilateur

16. Que fait runtime.GOMAXPROCS(n) ?

- A. Définit le nombre de goroutines simultanées
- B. Définit le nombre de processeurs logiques utilisés par le scheduler
- C. Arrête une goroutine
- D. Réinitialise toutes les goroutines

17. Quel type de fonction est adapté pour une goroutine ?

- A. Fonction bloquante avec for {}
- B. Fonction rapide sans interaction
- C. Fonction indépendante avec traitement parallèle
- D. Fonction sans return

18. Quelle fonction peut être utilisée pour identifier la goroutine actuelle ?

- A. goroutine.ID()
- B. debug.GoroutineID()
- C. runtime.GoID()

- D. Il n’y a pas de fonction officielle

19. Un WaitGroup peut être réutilisé :

- A. Oui, après avoir appelé Wait()
- B. Non, il est à usage unique
- C. Seulement avec un Mutex
- D. Seulement dans le main()

20. Lequel est vrai concernant go ?

- A. Il attend que la fonction appelée se termine
- B. Il exécute immédiatement la fonction appelée
- C. Il planifie la fonction pour exécution concurrente
- D. Il exécute dans le même thread que main()

21. Peut-on utiliser un WaitGroup dans une boucle ?

- A. Non
- B. Oui, mais avec précaution
- C. Oui, sans limite
- D. Seulement avec des mutex

22. Que se passe-t-il si on appelle wg.Done() sans Add() ?

- A. Le programme panique
- B. La goroutine est ignorée
- C. Wait() n’est jamais débloqué
- D. Rien de spécial

23. Peut-on utiliser un Mutex pour synchroniser une lecture ?

- A. Oui, mais c’est inutile
- B. Non, seulement pour les écritures
- C. Oui, pour éviter les *data races* même en lecture
- D. Uniquement dans main()

24. Le scheduler Go :

- A. Utilise un thread par goroutine
- B. Alloue dynamiquement des threads système aux goroutines
- C. Est manuel

- D. N'existe pas

25. Quelle est la bonne séquence pour utiliser un WaitGroup ?

- A. Add(), Wait(), Done()
- B. Wait(), Add(), Done()
- C. Add(), Done(), Wait()
- D. Add(), Done() uniquement

26. Peut-on imbriquer des goroutines ?

- A. Non
- B. Oui, mais cela bloque main()
- C. Oui, mais c'est interdit en production
- D. Oui, une goroutine peut lancer une autre goroutine

27. Que fait un Mutex.Lock() si la ressource est déjà verrouillée ?

- A. Lève une exception
- B. Retourne false
- C. Bloque jusqu'à libération
- D. Ignore la requête

28. Que permet sync.Once ?

- A. Créer une seule goroutine
- B. Verrouiller une seule variable
- C. Exécuter une fonction **exactement une fois**, même avec plusieurs goroutines
- D. Répéter une opération plusieurs fois

29. Quelle est la sortie probable de ce code ?

for i := 0; i < 3; i++ { go fmt.Println(i)}

- A. 0 1 2
- B. 2 2 2
- C. Sortie imprévisible
- D. Panique du programme

30. Pourquoi utiliser defer wg.Done() au début de la goroutine ?

- A. Pour ralentir la goroutine
- B. Pour s'assurer que Done() est toujours appelé

- C. Pour éviter un *panic*
- D. Pour améliorer la performance