Technical Assessment Test – Golang

Introduction:

Welcome to the Golang Technical Assessment! This test evaluates your proficiency in Golang, covering language fundamentals and advanced concepts. It includes multiple-choice questions, coding exercises, and open-ended questions.

Rules:

No External Tools: Strictly refrain from using external tools, chat platforms, or equivalents.

Code Integrity: Write clear, concise, and well-documented code adhering to Golang best practices.

Honesty and Integrity: Submit only your original work to avoid disqualification.

Best of luck! Show us your Golang expertise ! 🚀

Section 1: Multiple Choice Questions

1. What is Golang's primary focus in terms of development?

a. Web development

b. System programming

c. Mobile application development

d. Data analysis

2. Which of the following is a correct way to declare a slice in Golang?

a. `var s []int`

b. `s := make([]int, 0)`

c. `s := []int{}`

d. All of the above

3. In Golang, what does the `defer` keyword do?

a. Defer execution until the end of the program

b. Defer execution until the end of the enclosing function

c. Execute immediately

d. None of the above

4. What is the purpose of the `init` function in Golang?

a. It is called before the main function

b. It is called after the main function

c. It is called when a package is imported

d. It is used for garbage collection

5. Which of the following is true about goroutines in Golang?

a. They are lightweight threads managed by the Go runtime

b. They are only suitable for CPU-bound tasks

c. They are not supported in Golang

d. They require explicit memory management

6. What is the correct way to create a new instance of a struct in Golang?

a. `new(MyStruct)`

b. `MyStruct{}`

c. `&MyStruct{}`

d. Both b and c

7. In Golang, what does the `make` function do when used with a channel?

a. Creates a new channel

b. Allocates and initializes a channel

c. Closes the channel

d. Sends a value on the channel

8. How do you handle errors in Golang?

a. Use panic and recover

b. Check for errors and return them

c. Ignore errors for simplicity

d. Errors are automatically handled by the runtime

9. What is the purpose of the `context` package in Golang?

a. It provides a way to cancel operations

b. It is used for defining constants

c. It handles HTTP requests

d. It manages database connections

10. How do you declare and initialize a map in Golang?

a. `var m map[string]int `

b. `m := map[string]int{} `

c. `m := make(map[string]int) `

d. All of the above

Section 2: Code Production

11. Database Interaction

Integrate a simple in-memory database.
Create a struct to represent a Product with fields: ID, Name, and Price.
Implement an endpoint /api/products that returns a JSON array of sample products.

```
// Your code here
```

12. Implement a Golang REST API endpoint for creating a new user. The user data should be sent as JSON in the request body, and the API should return the created user's details.

```
// Your code here
```

13. Write a Golang function that checks if a given string is a palindrome.
Explain your approach.

```
func isPalindrome(s string) bool {
    // Your explanation and code here
}
```

14. Implement a Golang program that concurrently fetches data from multiple URLs and aggregates the results. Explain how you handle concurrent operations.

```
// Your code here
```

Section 3: Code Analysis

15. Analyze the following code. What will be the output, and can you identify any potential issues or improvements?

```go
package main

import "fmt"

func main() {
    s1 := []int{1, 2, 3}
    s2 := s1[:2]
    s2[0] = 4
    fmt.Println(s1[0])
}
```

16. Consider the following code. Explain the purpose of the done channel and any potential issues you may foresee.

```go

func main() {
    ch := make(chan int)
    done := make(chan bool)

    go func() {
        // Some time-consuming task
        ch <- 1
        done <- true
    }()

    // Your explanation and code here

    <-done
    close(ch)
}
```

17. In Golang, how would you prevent data races in concurrent programs? Provide an example or explain your approach.

18. Describe the role of the `` `http.HandleFunc` `` function in Golang and how it contributes to building a web server.

19. Review the code snippet below. Explain the purpose of the `` `defer` `` statement in the `` `someFunction` `` function.

```go
func someFunction() {
    defer fmt.Println("World")
    fmt.Print("Hello, ")
}
```

20. In Golang, what is the purpose of the `` `json` `` package, and how would you use it to handle JSON data?

Section 4: Golang Beyond the Basics

21. Explain the concept of interfaces in Golang and provide an example of how you would use them in a program.

22. Discuss the significance of the `` `context` `` package in the context of handling timeouts and cancellations in Golang applications.

23. Elaborate on the principles of error handling in Golang. How do you design a robust error-handling strategy in a large-scale application?

24. Describe the characteristics and use cases of the `` `sync.WaitGroup` `` type in Golang. Provide an example of its usage in concurrent programming.

25. Discuss the benefits and potential challenges of using goroutines in Golang for concurrent programming. How would you manage synchronization between goroutines?