***Java Final Assessment – Answers***

***Theoretical Questions:***

**Q1. What’s the difference between a goroutine and an operating system thread?**

**Goroutine:** A Goroutineis a function or method which executes independently and simultaneously in connection with any other Goroutines present in your program. Or in other words, every concurrently executing activity in Go language is known as a Goroutines.

**Thread:** A process is a part of an operating system which is responsible for executing an application. Every program that executes on your system is a process and to run the code inside the application a process uses a term known as a thread. A thread is a lightweight process, or in other words, a thread is a unit which executes the code under the program. So every program has logic and a thread is responsible for executing this logic.

Here are some of the differences between Goroutine and Thread:

|  |  |
| --- | --- |
| **Goroutine** | **Thread** |
| Goroutines are managed by the go runtime. | Operating system threads are managed by kernal. |
| Goroutine are not hardware dependent. | Threads are hardware dependent. |
| Goroutines have easy communication medium known as channel. | Thread does not have easy communication medium. |
| Due to the presence of channel one goroutine can communicate with other goroutine with low latency. | Due to lack of easy communication medium inter-threads communicate takes place with high latency. |
| Goroutine does not have ID because go does not have Thread Local Storage. | Threads have their own unique ID because they have Thread Local Storage. |
| Goroutines are cheaper than threads. | The cost of threads are higher than goroutine. |
| They are cooperatively scheduled. | They are preemptively scheduled. |
| They have fasted startup time than threads. | They have slow startup time than goroutines. |
| Goroutine has growable segmented stacks. | Threads does not have growable segmented stacks. |

**Q2. Can constants be computed in Go?**

### **Constants in Go:**

* Must be able to be assigned at compile time. The value of a const can’t be the result of a runtime calculation
* Run faster because the compiler can make specific optimizations
* Cannot change. The compiler will not allow them to be re-assigned
* Only work with some types. [Arrays, Slices, Maps, Structs, etc… can’t be made constant (or can they?)](https://qvault.io/2019/10/21/golang-constant-maps-slices/)
* Are not normal Go types unless explicitly assigned as such.

In Go, constants provide *complete safety* in regards to the value they hold. They cannot be computed (making them used less often), but are guaranteed to always reference the same value.

**Q3. What does the Go ecosystem use for package and dependency management?**

Any folder that contains a Go source file is a Go package. Its name is the path of the folder relative to $GOPATH/src. For example, a folder containing Go source files located at $GOPATH/src/github.com/alice/foo is named github.com/alice/foo.

**Q4. How would you succinctly swap the values of two variables in Go?**

Here, we will learn the easiest (succinctly) way to swap the values of two variables in Golang.

The values can be swapped by using the following way,

x, y = y, x

**Example:**

**package** main

**import** (

"fmt"

)

**func** main() {

x := **10**

y := **20**

fmt.Printf("Before swapping, x: %d, y: %d\n", x, y)

x, y = y, x

fmt.Printf("After swapping, x: %d, y: %d\n", x, y)

}

**Output:**

Before swapping, x: 10, y: 20

After swapping, x: 20, y: 10

**Q5. Do you have any preferences for error handling methodologies in Go?**

We can take advantage of multiple return values feature by adding an error struct to returned values. **By convention, an error value will be on the right and a result value will be on the left.** This convention kind of bugs me because it doesn’t sound right that the error is on the right side.

Func Marshal(v interface{}) ([]byte, error) {  
e := &encodeState{}  
err := e.marshal(v, encOpts{escapeHTML: *true*})  
if err != nil {  
return nil, err  
}  
return e.Bytes(), nil

}

The above code snippet is taken from the json package in the standard library. Notice that the *error* is a type. It is an interface declared in the builtin package.

**Q6. What is a pointer and when would you use it?**

Pointers in Go programming language or Golang is a variable that is used to store the memory address of another variable. Pointers in Golang is also termed as the special variables. The variables are used to store some data at a particular memory address in the system. The memory address is always found in hexadecimal format(starting with 0x like 0xFFAAF etc.).

**Q7. Describe the difference between sync.Mutex and sync.RWMutex?**

A RWMutex is a reader/writer mutual exclusion lock. The lock can be held by an arbitrary number of readers or a single writer. The zero value for a RWMutex is an unlocked mutex. In other words, readers don't have to wait for each otherThey only have to wait for writers holding the lock. A sync.RWMutex is thus preferable for data that is mostly read, and the resource that is saved compared to a sync.Mutex is time.

**Q8. Consider the following code. What will be the value of s1?**

The code is not given in the question. Question is incomplete.

**Q9. Are channels and maps safe for concurrent access?**

Maps are not safe for concurrent use: it's not defined what happens when you read and write to them simultaneously. If you need to read from and write to a map from concurrently executing goroutines, the accesses must be mediated by some kind of synchronization mechanism.

**Q10. How would you sort a slice of custom structs?**

GO has a sort package that provides utility primitives for the sorting of slices and user-defined types. Any collection can be sorted by the Sort function of sort package of GO it if implements the sort.Interface.

***Practical Questions:***

**Q1. How to swap two numbers without using a third variable?**

**Step 1**: Define a function that accepts two numbers, type is int.

**Step 2**: Find b = a + b;

**Step 3**: Then a = b – a and b = b – a

Package main

Import “fmt”

Func swap (a, b int ){

fmt.Println(“Before swapping, numbers are %d and %d\n”, a, b)

b = a + b

a = b – a

b = b – a

fmt.Println(“After swapping, numbers are %d and %d\n”, a, b)

}

Func main(){

Swap (23, 45)

Swap (56, 100)

}

**Q2. Golang Program to check if a vowel is present in the string?**

package main

import "fmt"

func main() {

var user\_string string

var check\_for\_vowel bool

fmt.Print("Enter a String to check if vowel is present or not:")

fmt.Scanf("%s\n", &user\_string)

for \_, char := range user\_string {

if char == 'a' || char == 'e' || char == 'i' || char == 'o' || char == 'u' {

check\_for\_vowel = true

}

}

if check\_for\_vowel {

fmt.Println("The String Contains vowel.")

} else {

fmt.Println("There are No vowel in the String.")

}

}

**Q3. Golang program to check if the given number is Prime?**

package main

import "fmt"

func CheckPrime(number int) {

isPrime := true

if number == 0 || number == 1 {

fmt.Printf(" %d is not a  prime number\n", number)

} else {

for i := 2; i <= number/2; i++ {

if number%i == 0 {

fmt.Printf(" %d is not a  prime number\n", number)

isPrime = false

break

}

}

if isPrime == true {

fmt.Printf(" %d is a prime number\n", number)

}

}

}

func main() {

var user\_input int

fmt.Print("Enter a String to check if it is prime a number or not:")

fmt.Scanf("%s\n", &user\_input)

CheckPrime(user\_input)

}

}

**Q4. Please write a Golang program to add tab, new line and print "Mr. ABC" along with double quotes from a string?**

package main

import "fmt"

func main() {

fmt.Println("\n\t\"Mr. ABC\"")

}

**Q5. Please write a Golang program to find sum and average from the given array [2,4,6,8,34,5,7,3,5,67,4]**

package main

import "fmt"

func main() {

// declaring an array of values

array := []int{2, 4, 6, 8, 34, 5, 7, 3, 5, 67, 4}

// size of the array

n := 11

// declaring a variable

// to store the sum

sum := 0

// traversing through the

// array using for loop

for i := 0; i < n; i++ {

// adding the values of

// array to the variable sum

sum += (array[i])

}

// declaring a variable

// avg to find the average

avg := (float64(sum)) / (float64(n))

// typecast all values to float

// to get the correct result

fmt.Println("Sum = ", sum, "\nAverage = ", avg)

}

**Q6. Please write a program to find factorial of the number input from the keyboard?**

package main

import "fmt"

func factorial(x uint) uint {

if x == 0 {

return 1

}

return x \* factorial(x-1)

}

func main() {

var user\_input uint

fmt.Print("Enter a Number to know its Factorial:")

fmt.Scanf("%s\n", &user\_input)

println(factorial(user\_input))

}

**Q7. Create a Pyramid of Characters in in Golang?**

package main

import "fmt"

func main() {

var rows int = 5

var k int

for i := 1; i <= rows; i++ {

k = 0

for space := 1; space <= rows-i; space++ {

fmt.Print("  ")

}

for {

fmt.Print("\* ")

k++

if k == 2\*i-1 {

break

}

}

fmt.Println("")

}

}

**Q8. Please write a program to concatenate two given arrays [3,5,76,3,6,3,5,6,3] and [2,3,65,7,4,3,6,3,56,3]?**

package main

import "fmt"

func main() {

var slice\_1 = []int{3, 5, 76, 3, 6, 3, 5, 6, 3}

var slice\_2 = []int{2, 3, 65, 7, 4, 3, 6, 3, 56, 3}

slice\_3 := append(slice\_1, slice\_2...)

fmt.Printf("slice\_1: %v\n", slice\_1)

fmt.Printf("slice\_2: %v\n", slice\_2)

fmt.Printf("Concantination on slice\_1 and slice\_2: %v\n", slice\_3)

}

**Q9. Find second largest number in an array [3,5,76,3,6,3,5,6,3]?**

package main

import "fmt"

func main() {

var large1 int = 0

var large2 int = 0

arr := [...]int{3, 6, 76, 7, 6, 55, 55, 6, 77}

large1 = arr[0]

for i := 1; i <= 4; i++ {

if large1 < arr[i] {

large2 = large1

large1 = arr[i]

} else if large2 < arr[i] {

large2 = arr[i]

}

}

fmt.Println("Second largest element is: ", large2)

}

**Q10. Please write a Golang program to print your name in the text file?**

package main

import (

"fmt"

"io/ioutil"

"log"

"os"

)

func CreateFile() {

file, err := os.Create("test.txt") // Truncates if file already exists, be careful!

if err != nil {

log.Fatalf("failed creating file: %s", err)

}

defer file.Close() // Make sure to close the file when you're done

len, err := file.WriteString("Sudeb Dolui")

if err != nil {

log.Fatalf("failed writing to file: %s", err)

}

fmt.Printf("\nLength: %d bytes", len)

fmt.Printf("\nFile Name: %s", file.Name())

}

func main() {

fmt.Printf("########Create a file and Write the content #########\n")

CreateFile()

}