Week 7 Assessment

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

Ans.)

|  |  |
| --- | --- |
| 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?

Ans.) 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?

Ans.) Go modules or go.mod is used which is one of many package managers to deal with dependencies in Go.

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

Ans.) In the program if we add the line

x, y := y, x  
 Thinking that x and y are our declared variables. Then we can succinctly swap the values

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

Ans.) Defer, panic and recover are the methods used to handle errors in GoLang.

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

Ans.) Pointers in Go programming language or [Golang](https://www.geeksforgeeks.org/go-programming-language-introduction/)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](https://www.geeksforgeeks.org/go-variables/) are used to store some data at a particular memory address in the system. A pointer is a special kind of variable that is not only used to store the memory addresses of other variables but also points where the memory is located and provides ways to find out the value stored at that memory location. It is generally termed as a Special kind of Variable because it is almost declared as a variable but with \*(dereferencing operator).

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

Ans.)

Mutex:

1. There are two operations for mutex: acquire lock and release lock.
2. When a goroutine acquires the mutex, no goroutine can acquire the mutex. You can only wait for the goroutine to release the mutex.
3. Mutex is applicable to the situation that the number of read and write operations is almost the same.
4. Both reading and writing can be put into the mutex

Read write lock:

1. There are four operations of read lock, read lock, read unlock, write lock and write unlock.
2. There can be at most one write lock and multiple read locks (the maximum data is related to the number of CPU s).
3. The priority of the write lock is higher than that of the read lock. This is because the write lock has been blocked to prevent too many read locks.
4. When one goroutine obtains a write lock, other goroutines cannot obtain a read lock or a write lock until the write lock is released.
5. When one goroutine gets a read lock, other goroutines can get a read lock, but not a write lock. Therefore, it can also be seen that if a goroutine wants to acquire a write lock, other goroutines will constantly acquire and release the read lock, which will cause the write lock to be blocked all the time, so it can be avoided that the priority of the write lock is higher than that of the read lock.
6. The read-write lock is applicable to the situation of more reading and less writing.
7. When there are many read operations, for example, there are three goroutines: G1, G2 and G3, all of which want to read A piece of data A. if we use the mutex lock, it is the following situation: G1 first locks, then reads A, and then releases; G2 then locks, reads A, and releases; G3 locks, reads A, and then releases This operation is serial. Since each goroutine needs to wait in line for the previous goroutine to release the lock, the efficiency is obviously not high. However, if we use the read-write lock at this time, we can make G1, G2, G3 read A at the same time, which can greatly improve the efficiency.
8. The write operation can only be placed in the write lock, and the read operation can be placed in the read-write lock, but the concurrency efficiency must be low in the write lock.

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

Ans.) Code Missing.

Q9. Are channels and maps safe for concurrent access?

Ans.) Maps in go are not safe for concurrency. You must take a lock to read and write on them if you will be accessing them concurrently. While Channels are safe as they run concurrently.

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

Ans.) 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?

Ans.) Program:

package main

import "fmt"

func main() {

    x, y := 10, 20

    fmt.Println("Before swapping x:", x, "y:", y)

    x, y = y, x

    fmt.Println("Before swapping x:", x, "y:", y)

}

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

Ans.) Program:

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?

Ans.)Program:

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 vowel is present 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?

Ans.) Program:

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]

Ans.) Program:

import "fmt"

func main() {

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

    sum := 0

    for i := 0; i < len(array); i++ {

        sum = sum + array[i]

    }

    fmt.Println(sum)

}

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

Ans.) Program:

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 String to check if vowel is present or not:")

    fmt.Scanf("%s\n", &user\_input)

    println(factorial(user\_input))

}

Q7. Create a Pyramid of Characters in in Golang?

Ans.) Program:

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]?

Ans.) Program:

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]?

Ans.) Program:

package main

import "fmt"

func main() {

    var large1 int = 0

    var large2 int = 0

    arr := [...]int{3, 5, 76, 3, 6, 3, 5, 6, 3}

    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?

Ans.) Program:

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()

}