Go Training Pathway

Coding Guidelines

While working on the assignments, refer to the common review comments made during reviews of Go code – <https://github.com/golang/go/wiki/CodeReviewComments>

Assignments

1. **Frequency Problem**
2. Accept a series of digits and count the frequency of each digit using integer slice.

eg: 12349120

Output – 1 2 2 1 1 0 0 0 0 1

Conditions –

1. Accept only numerals
2. Input to have maximum 15 digits
3. Optimize the use of memory allocated to the slice.
4. Accept a series of words and count its frequency using a map of type [string]int

eg: go bat cat gopher lang go toy go

go : 3

bat: 1

cat: 1

gopher: 1

lang: 1

toy: 1

Conditions –

1. Accept only alphabets.
2. Input to be considered case insensitive viz. GO, go and Go are same words.

1. **Unit Testing in Go**

Write unit test cases for the modules in the frequency problem. Here you would need to make the code more modular so as to be unit tested. Unit tests should cover the full functionality of the problem(Assignment 1).

1. **Web Crawling Problem**

Write a program to web crawl and fetch the data from 3 different websites –

<https://www.python.org/>, <https://www.ruby-lang.org/en/>, and <https://golang.org/> and do the following –

1. Count the number of bytes read. (HINT: use io.Copy(ioutil.Discard, data))
2. Time taken to crawl each site.
3. Count the total time taken to crawl the data from all the sites. (HINT: Start timer before starting to load the data and ends it when all the crawling is done.)

type Lang struct{

Name string

URL string

bytes uint

time Time.Seconds

}

Write a function with prototype *crawl(pfunc func(Lang), lang Lang,)*to take a lambda function as input which prints the read data in the following formats –

1. Go formatted output using %v
2. JSON formatted

1. **Concurrent Web Crawling**

Note that in the above example -

*TotalCrawlTime = CrawlTime(website1) + CrawlTime(website2) + CrawlTime(website3)*

Improve the web crawling in the previous example by making it concurrent using \*sync.WaitGroup to wait for all the concurrent goroutines to finish.

1. **Goroutines and Channels**

Modify the above program to count the total number of bytes read by the program from the 3 sites and also the cumulative actual time taken to fetch data.

1. **Interfaces in Go**
2. Create an integer enum in Go and print its string representation using fmt.Printf(“%s”, Eval), where Eval is a constant of that enum type.
3. For the following interface, create 4 different implementations along with their unit tests.

type Planet interface{

Name() string

Mass() int

}

Also, define proper methods to print the name of the planet when running –

fmt.Printf(“%s”, pluto), where pluto is an object implementing the Planet interface.

1. **TCP Client Server**

Write a TCP client server program in go to listen on a specific port (assume any). Running both, user should be able to interact in the following manner –

Client Request – Hello

Server Response – Hi

Client Request – Name

Server Response – Chitty.

Client Request – GoodBye

Server Response – Bye.

Note – Server should be kept running all the time to be able to handle request from the clients.

Also, write unit test cases for the both client and server. Use mocking logic writing the unit tests.