Net-centric Programming

Lab: Goroutines and Channels

I. Objective:

- Get familiar with Go Programming Concurrency
- Sample codes using Goroutines:

```
package main

import (
	"fmt"
	"strconv"
)

func main() {
	ch := make(chan string)

	for i := 0; i < 10; i++ {
		go func(i int) {
			for j := 0; j < 10; j++ {
				ch <- "Goroutine : " + strconv.Itoa(i)
			}
		}(i)
	}

	for k := 1; k <= 100; k++ {
		fmt.Println(k, <-ch)
	}
}
//This go program init 10 goroutines to write value to a shared channel
```

```
package main

import (
	"fmt"
)

func main() {
	var intSlice = []int{42, 17, 89, 74, 5, 63, 38, 49, 92, 29}
	chOdd := make(chan int)
	chEven := make(chan int)

	go odd(chOdd)
	go even(chEven)

	for _, value := range intSlice {
		if value%2 != 0 {
			chOdd <- value
		} else {
			chEven <- value
		}
	}

}

func odd(ch <-chan int) {
	for v := range ch {
		fmt.Println("ODD :", v)
	}
}

func even(ch <-chan int) {
	for v := range ch {
		fmt.Println("EVEN:", v)
	}
}
//This program init 2 goroutines. These two program checks odd and even number of a given
series of integers
```

```
package main

import (
	"fmt"
	"sync"
	"time"
)

func main() {
	var wg sync.WaitGroup

	// Add the number of goroutines to wait for
	wg.Add(3)

	go performTask("Task 1", &wg)
	go performTask("Task 2", &wg)
	go performTask("Task 3", &wg)

	// Wait for all goroutines to complete
	wg.Wait()

	fmt.Println("All tasks completed.")
}

func performTask(taskName string, wg *sync.WaitGroup) {
	defer wg.Done()

	fmt.Printf("Starting %s\n", taskName)
	time.Sleep(2 * time.Second) // Simulate some work
	fmt.Printf("Completed %s\n", taskName)
}
// This go program use WaitGroup to synchroinze goroutines
```

II. Questions:

1. (Character Frequency) Find the occurrence of each character in a string. Improve this code using concurrency. Perform this task on a text file (optional)

```
Example: Net-centric Programming
n: 3
e: 2
t: 2
-: 1
r: 3
i: 2
c: 2
(blank): 1
p: 1
g: 2
a: 1
m: 2
```

2. Given the case study of International University library:

- The library can seat up to 30 students at a given time
- Usually, there is around 100 students vist everyday
- For each vist, students spend from 1 to 4 hours reading at the lib

Write a go program to simuate the library operation in one day:

- Randomly generate and assign "reader ID" to student that come to the lib.
- Use sleep to simulate the time (1 second = 1 hour in real life)
- First come first serve. If the lib is full, student need to wait
- After finish studying at the lib, student will leave
- How many hours the lib need to open to serve all the students

Example:

```
Time 0: Student 3 start reading at the lib
Time 0: Student 15 start reading at the lib
Time 0: Student 2 start reading at the lib
…
Time 1: Student 45 is waiting
….
Time 5: Student 5 is leaving. Spent 3 hours reading
…
Time 10: No more students. Lets call it a day
```

The end.