# Matchmove Golang Test Assessment

1. Create a small program which accepts two slices of colors. It must return a slice containing the colors that appear in **either or both** slices.

```
c1 := []string{"Red", "Black", "White"}

c2 := []string{"Black", "Yellow", "Orange"}
```

should return in any order a slice which should have no duplicates.:

```
[Red Black White Yellow Orange]
```

2. Create a small program with a for loop which can iterate over the sequential values sent on the **channel** until it closes.

```
Values: 10, 20, 35, 100, 200, 502
```

3. The program below does have an issue when being executed. Output will be different for each time it's executed because of the **race condition**. To resolve the issue please make use of **mutex**.

```
package main

import (
	"fmt"
	"sync"
)

var x = 0

func increment(wg *sync.WaitGroup) {
	x = x + 1
	wg.Done()
}

func main() {
	var w sync.WaitGroup
	for i := 0; i < 1000; i++ {
		w.Add(1)
		go increment(&w)
	}
	w.Wait()
	fmt.Println("final value of x", x)
}
```

4. Based on the program codes above given on problem number 3, resolve the issue using **channels**.

5. Write unit tests for the program below. Test the **Subtract** and the **Add** functions. Make use of **structs** and **slices** to satisfy the following operations with sets of numbers:

- For Addition:
    - 1 and 2
    - 3 and 4
    - 5 and 6
- For Subtraction:
    - 2 and 1
    - 4 and 3
    - 6 and 5

```
package math

func Subtract(x, y int) (res int) {
	return x - y
}

func Add(x, y int) (res int) {
	return x + y
}
```

6. Write a small program which simulates capturing of the status code for errors produced by an HTTP request. The objective is to come up with custom errors to capture detailed error information cleanly. Output must be like the ones below for 503 status code:

```
status 503: err unavailable
exit status 1
```