PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

INF239 SISTEMAS OPERATIVOS
Semestre 2024-2
Laboratorio 4

1) (5 puntos – nombre del programa: *ejer1.go*) Se tiene el siguiente programa en Go:

```
 1  package main
 2
 3  import (
 4          "fmt"
 5          "sync"
 6  )
 7
 8  var wg sync.WaitGroup
 9
10  func worker1() {
11          for x := 0; x < 10; x++ {
12                  fmt.Printf("I like to ")
13                  fmt.Println("systems")
14          }
15          wg.Done()
16  }
17
18  func worker2() {
19          for x := 0; x < 10; x++ {
20                  fmt.Printf("study ")
21                  fmt.Printf("operating ")
22          }
23          wg.Done()
24  }
25
26  func main() {
27          wg.Add(2)
28          go worker1()
29          go worker2()
30          wg.Wait()
31  }
```

Al ejecutarlo, después de compilarlo, se obtiene:

```
study operating study operating study operating study operating study operating
study operating study operating study operating study operating study operating
I like to systems
I like to systems
I like to systems
I like to systems
I like to systems
I like to systems
I like to systems
I like to systems
I like to systems
I like to systems
```

Sincronice el programa para que imprima 10 líneas con el mensaje:

```
I like to study operating system
I like to study operating system
. . .
```

Restricciones

a) Los mensajes no deben ser modificados. Puede agregar, pero no eliminar líneas.
b) No puede emplear ninguno tipo de canales, ya sea síncronos o asíncronos.

2) (5 puntos – nombre del programa: *ejer2.go*) Se tiene el siguiente programa en Go:

```
 1  package main
 2
 3  import (
 4          "fmt"
 5          "sync"
 6  )
 7
 8  var wg sync.WaitGroup
 9
10  func A() {
11          for x := 0; x < 100; x++ {
12                  fmt.Printf("a1")
13                  fmt.Printf("a2")
14          }
15          wg.Done()
16  }
17
18  func B() {
19          for x := 0; x < 100; x++ {
20                  fmt.Printf("b1")
21                  fmt.Printf("b2")
22          }
23          wg.Done()
24  }
25
26  func main() {
27          wg.Add(2)
28          go A()
29          go B()
30          wg.Wait()
31          fmt.Println()
32  }
```

Al ejecutarlo, después de compilarlo, se obtiene:

```
b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1a1a2a1a2a1a2a1a2a1b2b1b2b1b2b1b2b1b2b1b
2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2
b1b2b1b2b1b2b1b2b1b2b1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a
1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1
a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a
2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1
b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2a1a2a1a2a1a2b1b2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a
2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2a1a2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2
b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2b1b2
```

Sincronice el programa para que siempre imprima secuencias válidas. Solo son válidas las siguientes secuencias

`a1b1a2b2` `b1a1a2b2` `b1a1b2a2`

Restricciones

a) Los mensajes no deben ser modificados. Puede agregar, pero no eliminar líneas.
b) La secuencia no debe ser única, tampoco debe de existir alternancia estricta entre las secuencias.
c) Solo puede hacer uso de canales, ya sea síncronos o asíncronos. No puede hacer uso de variables tipo `sync.Mutex` (`Lock`, `Unlock`).

3) (10 puntos – nombre del programa: *ejer3.go*) El problema de la cena de los salvajes. Una tribu de salvajes tiene una cena en comunidad. La comida se toma de una gran olla que puede contener 5 porciones de estofado. Cuando un salvaje quiere comer, él mismo debe tomar una porción de la olla, a menos que esté vacía. Si la olla está vacía, el salvaje despierta al cocinero y luego espera hasta que el cocinero haya surtido nuevamente la olla con 5 porciones.

El siguiente código muestra 11 *gorutinas* que simulan 10 salvajes y un cocinero. El código no está sincronizado.

```
 1  package main
 2
 3  import (
 4          "fmt"
 5          "math/rand"
 6          "sync"
 7          "time"
 8  )
 9
10  var (
11          wg sync.WaitGroup
12          t  int
13  )
14
15  func cook() {
16          for {
17                  putServingsInPot(5)
18          }
19          wg.Done()
20  }
21
22  func putServingsInPot(n int) {
23          fmt.Printf("Put Servings %d In Pot\n", n)
24          t = rand.Intn(10) // n will be between 0 and 10
25          time.Sleep(time.Duration(t) * time.Millisecond)
26  }
27
28  func getServingFromPot(n int) {
29          fmt.Printf("%d: Get serving from Pot\n", n)
30  }
31
32  func eat(n int) {
33          fmt.Printf("%d: eating...\n", n)
34  }
35
36  func savage(n int) {
37          for {
38                  getServingFromPot(n)
39                  eat(n)
40                  t = rand.Intn(10) // n will be between 0 and 10
41                  time.Sleep(time.Duration(t) * time.Millisecond)
42          }
43          wg.Done()
44  }
45
46  func main() {
47          rand.Seed(time.Now().UnixNano())
48          wg.Add(11)
49          go cook()
50          for x := 0; x < 10; x++ {
51                  go savage(x)
52          }
53          wg.Wait()
54  }
```

Este programa al ser ejecutado obtiene la siguiente salida:

```
alejandro@abdebian:2022_2_lab3_SO$ ./ejer3
Put Servings 5 In Pot
2: Get serving from Pot
2: eating...
1: Get serving from Pot
1: eating...
8: Get serving from Pot
8: eating...
0: Get serving from Pot
0: eating...
7: Get serving from Pot
6: Get serving from Pot
6: eating...
3: Get serving from Pot
3: eating...
4: Get serving from Pot
4: eating...
9: Get serving from Pot
9: eating...
7: eating...
5: Get serving from Pot
5: eating...
```

Como se puede observar las salidas no están sincronizadas. Se le solicita sincronizar las *gorutinas* de a cuerdo a las siguientes restricciones:

- Los salvajes no pueden invocar `getServingFromPot` si la olla está vacía
- El cocinero puede invocar `putServingsInPot` solo si la olla está vacía

No hay más restricciones.

Porf. Alejandro T. Bello Ruiz