Distributed Communication 2nd practice

Li Jianhao
lijianhao288@hotmail.com

1 Basics

1.1 Defer

Syntax: defer <function call>

```
package main
import "fmt"
func main(){
    fmt.Println("First")
    defer fmt.Println("Last")
    fmt.Println("Second")
}
```

Listing 1: defer1

```
First
Second
Last
```

```
conn, err := amqp.Dial("amqp://guest:guest@localhost:5672/")
if err != nil {
    panic(err)
}
defer conn.Close()
```

Listing 2: defer2

```
package main
import "fmt"
func main(){
    fmt.Println("First")
    defer fmt.Println("Last1")
    defer fmt.Println("Last2")
    defer fmt.Println("Last3")
    fmt.Println("Second")
}
```

Listing 3: multiple defer

```
First
Second
Last3
Last2
Last1
```

1.2 Recursion

3*4 = 4 + (4 + (4))

```
package main

import ("fmt")

func main(){
    fmt.Println(multiplicateR(3,4))
}

func multiplicateR (a int, b int) int{
    if a== 0{
        return 0
    }
    return b + multiplicateR(a-1, b)
}
```

Listing 4: recursion

```
12
```

hint: 3*4 = 3 + (3 + (3 + (3)))

1.3 For loop.

Syntax: for <initial>;<condition>;<step> { }
Syntax: for <condition> { }
Syntax: for { }

```
package main

import("fmt")

func main(){
    for i:=0; i<5 ;i++{
        fmt.Println("i",i)
    }

    j := 0
    for j<5 {
        fmt.Println("j",j)
        j++
    }
}
```

Listing 5: for loop

```
i 0
i 1
i 2
i 3
i 4
```

```
j 0
j 1
j 2
j 3
j 4
```

Syntax:break
Syntax:continue

```
package main

import("fmt")

func main(){
    fmt.Println("break")
    for i:=0; i<5 ;i++{
        if i == 3 {
            break
        }
        fmt.Println("i",i)
    }

    fmt.Println("continue")
    for i:=0; i<5 ;i++{
        if i == 3 {
            continue
        }
        fmt.Println("i",i)
    }
}
```

Listing 6: break and continue

```
break
i 0
i 1
i 2
continue
i 0
i 1
i 2
i 4
```

1.4 Slice. For Range.

Syntax: <SliceName> := [ ]<Type>{ <Elements>}
Syntax: <SliceName> = append(<SliceName>, <NewElement(s)>)
Syntax:
for <IndexName>, <ElementName> := range <SliceName> {
}
or

for _, <ElementName> := range <SliceName> {
}

```
package main
import "fmt"
func main() {
    animals := []string{
        "dog",
        "cat",
        "bird",
        "lion",
    }
    animals = append(animals, "panda")
    animals = append(animals, "tiger", "wolf")
    for index, animal := range animals {
        fmt.Println(index, animal)
    }
    for _, animal := range animals {
        fmt.Println(animal)
    }
}
```

Listing 7: Slice

```
0 dog
1 cat
2 bird
3 lion
4 panda
5 tiger
6 wolf
dog
cat
bird
lion
panda
tiger
wolf
```

1.5 Variadic Function

Syntax: func <Name>(<ParameterName> ...<Type>) (<Return types>) {<Function body> }

```
package main

import ("fmt")

func main(){
    result1 := product(2,3)
    fmt.Println(result1)

    result2 := product(2,3,4)
    fmt.Println(result2)
}
```

```
                                                                    12
func product(nums ...int) int {                                    13
    result := 1                                                    14
    for _, num := range nums {                                     15
        result *= num                                              16
    }                                                              17
    return result                                                  18
}                                                                  19
```

Listing 8: Variadic Function

```
6                                                                   1
24                                                                  2
```

hint:
result *= num
is the short way of
result = result * num

2 Practice

2.1 p1

Define a function named **fibonacci** which return the nth fibonacci number. (The Fibonacci sequence: each number is the sum of the two preceding ones, starting from 0 and 1. eg. 0,1,1,2,3,5,8,11,...)

fibonacci(4)= 2, fibonacci(5)= 3

In the main function, call the function **fibonacci** two times. First time pass it with 6. The second time pass it with 7. Print out the results.

(Hint(The hint will not appear during the exam): takes the first two fibonacci numbers as the terminal conditions. If n == 1, return the first fibonacci number 0. If n==2, return the second fibonacci number 1. Other fibonacci number equals the sum of the previous two fibonacci numbers(Here call the fibonacci function itself two times))

2.2 p2

Create a slice of type string, called **urls**. It has two intial elements: "www.google.com", "www.facebook.com".

2.3 p3

use a for loop to append "www.web<n>.com" as new elements.

$n \in [2, 8]$

(Hint(The hint will not appear during the exam): Use + to attach strings. Use strconv.itoa to convert the int to string. The int comes from the iterator.)

2.4 p4

Use the for range loop to print out all the elements in the slice **urls**. (print without the index).

2.5 p5

Define a variadic function named **sum** which return the sum of all the parameters. In the main function, call the function **sum** two times. First time pass it with 2 and 3. The second time pass it with 2, 3 and 4. Print out the results.