Ejercicio 1:

Hacer una funcion que reciba un parámetro p de tipo numérico y que devuelva la cantidad y la suma total de los valores v tales que 0 <= v <= p , siendo v un número divisible por 3 o por 5

```
package main

import "fmt"

func multiples(p int) (int,int){
	count,sum:=0,0
	for i:=0;i<=p;i++{
		if i%3==0 || i%5==0 {
			count++
			sum+=i
		}
	}
	return count,sum
}

func main() {
	fmt.Println(multiples(0))
	fmt.Println(multiples(1))
	fmt.Println(multiples(5))
	fmt.Println(multiples(9))
}
```

Ejercicio 2:

Modelar la funcionalidad para un sistema de cines que calcule los ingresos netos de una función en base a los asistentes y al precio base de la entrada. Existen tres tipos de asistentes y tienen las siguientes características:

- Normales: Pagan el 100%
- Jubilados: Tienen un 50% de descuento
- Invitados: No pagan nada

```go
package main

import (
	"fmt"
)

type Expectator interface{
	GetDiscount() float32
}

type BaseExpectator struct{}

func (*BaseExpectator) GetDiscount() float32{
	return 0
}

type Retired struct{
	BaseExpectator
}

func (*Retired) GetDiscount() float32{
	return 0.5
}

type Guess struct{
	BaseExpectator
}

func (*Guess) GetDiscount() float32{
	return 1
}

func netIncome(expectators []Expectator,ticketPrice float32) float32{
	var total float32

	for _,e := range expectators{
		total+=ticketPrice-e.GetDiscount()*ticketPrice
	}
	return total
}

func main() {
	expectators:=[...]Expectator{&Retired{},&BaseExpectator{},&Guess{}}
	fmt.Println(netIncome(expectators[:],100))
}
```