




# Ejercicio 1

•

Con el tablero y las bolitas de Gobstones podemos representar muchos objetos. En este caso queremos hacer una escalera  como esta:

	0	1	2	
4	1		1	4
3	1	1	1	3
2	1		1	2
1	1	1	1	1
0	1		1	0
	0	1	2	

Creá un programa que dibuje una escalera de color **Negro** . El cabezal empieza en el origen (o sea, en el borde Sur-Oeste) pero no te preocupes por dónde finaliza.

En este video te dejamos una posible resolución a este ejercicio:

[Modelo de examen] Ejercicio 1



```
1 program {
2   repeat(4){
3     Poner(Negro)
4     Mover(Norte)
5   }
6   Poner(Negro)
7   Mover(Sur)
8   Mover(Este)
9   Poner(Negro)
10  Mover(Este)
11  IrAlBorde(Norte)
12  repeat(4){
13    Poner(Negro)
14    Mover(Sur)
15  }
16  Poner(Negro)
17  Mover(Oeste)
18  Mover(Norte)
19  Poner(Negro)
20 }
```



▶ Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial

	0	1	2	
4				4
3				3
2				2
1				1
0				0
	0	1	2	

Tablero final

	0	1	2	
4	1		1	4
3	1	1	1	3
2	1		1	2
1	1	1	1	1
0	1		1	0
	0	1	2	

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





## Ejercicio 2

•

Queremos dibujar una mesa que quede bien con cualquier decoración 🍷. Para ello vamos a definir un procedimiento `DibujarMesa` que tome un color y dibuje una mesa. Por ejemplo, si lo invocáramos con `Rojo` haciendo `DibujarMesa(Rojo)`, la misma se vería así:

	0	1	2	3	
2	1	1	1	1	2
1	1			1	1
0	1			1	0
	0	1	2	3	

Definí el procedimiento `DibujarMesa` para que dibuje una mesa como la que te mostramos del `color` que reciba como argumento. El procedimiento debe dibujar desde el extremo Sur Oeste pero no te preocupes por dónde termina el cabezal.

En este video te dejamos una posible resolución a este ejercicio:

[Modelo de examen] Ejercicio 2



```
1 procedure DibujarMesa(color){  
2   Pata1(color)
```





```

3  Mover(Este)
4  Poner(color)
5  Mover(Este)
6  Poner(color)
7  Mover(Este)
8  Pata2(color)
9  }
10
11 procedure Pata1(color){
12     repeat(2){
13         Poner(color)
14         Mover(Norte)
15     }
16     Poner(color)
17 }
18 procedure Pata2(color){
19     repeat(2){
20         Poner(color)
21         Mover(Sur)
22     }
23     Poner(color)
24 }

```

▶ Enviar

## ✓ ¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:



Tablero inicial

	0	1	2	3	
2					2
1					1
0					0
	0	1	2	3	

Tablero final

	0	1	2	3	
2	1	1	1	1	2
1	1			1	1
0	1			1	0
	0	1	2	3	



Tablero inicial

	0	1	2	3	
2					2
1					1
0					0
	0	1	2	3	

Tablero final

	0	1	2	3	
2	1	1	1	1	2
1	1			1	1
0	1			1	0
	0	1	2	3	



Tablero inicial

	0	1	2	3	
2					2
1					1
0					0
	0	1	2	3	

Tablero final

	0	1	2	3	
2	1	1	1	1	2
1	1			1	1
0	1			1	0
	0	1	2	3	



Tablero inicial

	0	1	2	3	
2					2
1					1
0					0
	0	1	2	3	

Tablero final

	0	1	2	3	
2	1	1	1	1	2
1	1			1	1
0	1			1	0
	0	1	2	3	

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





# Ejercicio 3

Ahora vamos a definir una función `concatenacionEsIgual` que recibe tres strings y nos dice si la concatenación de los dos primeros es igual al tercero 🤖:

```
concatenacionEsIgual("mari", "posa", "langosta")
false

concatenacionEsIgual("guarda", "ropas", "guardaropas")
true
```

Definí la función `concatenacionEsIgual`.

En este video te dejamos una posible resolución a este ejercicio:

[Modelo de examen] Ejercicio 3



Solución

Consola

```
1 function concatenacionEsIgual(string1,string2,concatenacion){
2   return string1 + string2 === concatenacion
3 }
```





 Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





# Ejercicio 4

Ya sabemos que es muy importante saludar al llegar a un lugar, el problema está en que el saludo depende de la hora del día. ¡Por suerte este problema también se puede resolver programando! 🙌

Podríamos tener una función que dado un nombre y un horario retorne el saludo correcto. Si es antes de las 12 debería ser "Buenos días", si es después "Buenas tardes". Por ejemplo:

```
saludar("Luis", 11)
"Buenos días Luis"

saludar("Carolina", 12)
"Buenas tardes Carolina"

saludar("Rocío", 13)
"Buenas tardes Rocío"
```

Definí la función `saludar`.

En este video te dejamos una posible resolución a este ejercicio:

[Modelo de examen] Ejercicio 4



 Solución

&gt;\_ Consola

```
1 function saludar(nombre, hora){  
2   if (hora < 12) {  
3     return "Buenos días " + nombre;}  
4   else {return "Buenas tardes " + nombre;}  
5 }}
```

 Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





# Ejercicio 5

Vamos a sumarle (☺) funciones a una extravagante calculadora. Queremos obtener la suma de los elementos mayores a 7 de una lista. Por ejemplo:

```
sumaDeLosMayoresASiete([2, 5, 9, 8, 11])  
28 //Porque es la suma de 9, 8 y 11.
```

Definí la función `sumaDeLosMayoresASiete`.

En este video te dejamos una posible resolución a este ejercicio:

[Modelo de examen] Ejercicio 5



Solución >\_ Consola

```
1 function sumaDeLosMayoresASiete(numeros){  
2   let sumatoria = 0;  
3   for (let numero of numeros){  
4     if (numero > 7){  
5       sumatoria += numero  
6     }  
7   }  
8   return sumatoria;
```



9 }

 Enviar ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2021  Mumuki[Información importante](#)[Términos y Condiciones](#)[Reglas del Espacio de Consultas](#)



## Ejercicio 6

Una conocida aplicación para escuchar música online quiere hacer un resumen con la información importante de sus canciones 🎵. Las canciones se almacenan como registros de la siguiente forma:

```
let elGenioDeLaNada = {  
  nombre: "El genio de la nada",  
  banda: "Eruca Sativa",  
  duracion: 4  
};  
  
let lotusFlower = {  
  nombre: "Lotus flower",  
  banda: "Radiohead",  
  duracion: 5  
};
```

Queremos definir una función que retorne un resumen de manera simple. Por ejemplo:

```
🔗 resumenCancion(elGenioDeLaNada)  
"El genio de la nada de la banda Eruca Sativa tiene una duración de 240 segundos"  
  
🔗 resumenCancion(lotusFlower)  
"Lotus flower de la banda Radiohead tiene una duración de 300 segundos"
```

Definí la función `resumenCancion` que nos permita obtener el resumen solicitado.

En este video te dejamos una posible resolución a este ejercicio:

## [Modelo de examen] Ejercicio 6



💡 ¡Dame una pista!

 Solución  Consola

```
1 function resumenCancion(unaCancion){  
2   return unaCancion.nombre + " de la banda " + unaCancion.banda + "  
   tiene una duración de " + (unaCancion.duracion * 60) + " segundos";  
3 }
```



 Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

---

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)







# Ejercicio 7

¡Dejemos atrás a JavaScript para pasar a Ruby! 🍷

Vamos a tomar unos mates ☕, mejor aún ¡programémoslo! 😊

Para eso vamos a crear el objeto `AguaParaMate` para poder:

- calentarla una temperatura (si la calentamos 10 grados, su temperatura incrementa en 10);
- ver si tiene la temperatura correcta, es decir, si está a exactamente 80 grados.

Definí en Ruby, el objeto `AguaParaMate` que tenga un atributo `@temperatura` que inicialmente está en 0 con su getter. `AguaParaMate` entiende los mensajes `calentar_agua!` (que recibe la cantidad de grados a sumar por parámetro) y `temperatura_exacta?`.

En este video te dejamos una posible resolución a este ejercicio:

[Modelo de examen] Ejercicio 7



🔗 Solución

>\_ Consola

```
1 module AguaParaMate
2   @temperatura = 0
```





```
3
4  def self.temperatura
5      @temperatura
6  end
7
8  def self.calentar_agua!(grados)
9      @temperatura += grados
10 end
11
12 def self.temperatura_exacta?
13     @temperatura == 80
14 end
15 end
```

▶ Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





# Ejercicio 8

Que hermoso leer y transportarse a otros mundos. Por eso tenemos una [Biblioteca](#) con algunos libros. 📖

Teniendo en cuenta que los libros saben responder al mensaje [nombre](#) ...

Definí en Ruby el método [nombres\\_de\\_libros](#) que responda el nombre de los libros de la [Biblioteca](#).

En este video te dejamos una posible resolución a este ejercicio:

[Modelo de examen] Ejercicio 8



Solución

Consola


```
1 module Biblioteca
2   @libros = [Fundacion, Contacto, LaInsoportableLevedadDelSer,
3     Socorro, ComoAguaParaChocolate]
4   def self.nombres_de_libros
5     @libros.map { |libro| libro.nombre}
6   end
end
```



 Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





# Ejercicio 9

Una empresa de mudanzas, cuenta con varios transportes, para realizarlas elige un `Camion`.

Cada vez que un `Camion` carga un mueble, el mueble sufre una modificación:

- cada `Colchon` pierde 4 resortes;
- cada `Sillon` sube su `nivel_de_polvo` en 20;
- a los `Electrodomestico`s no les pasa nada.

Definí los metodos `cargar_muebles!` en la clase `Camion`. También definí el método `ser_cargado!` en los muebles junto con su respectivo getter.

En este video te dejamos una posible resolución a este ejercicio:

[Modelo de examen] Ejercicio 9



Solución

Consola

```
1 class Camion
2   def initialize(muebles)
3     @muebles = muebles
4   end
5   def cargar_muebles!
```



```
6      @muebles.each{|mueble|mueble.ser_cargado!}
7  end
8 end
9
10 class Colchon
11   def initialize(resortes)
12     @cantidad_de_resortes = resortes
13   end
14   def cantidad_de_resortes
15     @cantidad_de_resortes
16   end
17   def ser_cargado!
18     @cantidad_de_resortes -=4
19   end
20 end
21 class Sillon
22   def initialize(polvo)
23     @nivel_de_polvo = polvo
24   end
25   def nivel_de_polvo
26     @nivel_de_polvo
27   end
28   def ser_cargado!
29     @nivel_de_polvo +=20
30   end
31 end
32
33 class Electrodomestico
34   def ser_cargado!
35   end
36 end
```

▶ Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)

