

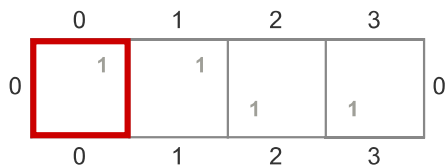


Te quedan 00:14:33

Ejercicio 1

•

Una fábrica de chocolates nos pidió un programa que se encargue de armar una caja de bombones con distintos sabores . Actualmente venden bombones de frutilla, menta y chocolate amargo que representaremos con bolitas de color Rojo, Verde y Negro respectivamente. Las cajas tienen cuatro bombones y esta en particular tendrá los siguientes gustos:



Es decir, una bolita de color Negro, al Este una de color Negro, al Este una de color Rojo y al Este una de color Rojo.

Creá el programa que haga la caja de bombones solicitada. El cabezal comienza en el extremo Sur Oeste y no importa dónde termina.

```

1 program {
2   Poner(Negro)
3   Mover(Este)
4   Poner(Negro)
5   Mover(Este)
6   Poner(Rojo)
7   Mover(Este)
8   Poner(Rojo)
9 }

```



Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial

	0	1	2	3	
0					0
	0	1	2	3	

Tablero final

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

Esta guía fue desarrollada por Mayra Mosqueira, Gustavo Trucco bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)





Te quedan 00:14:02

Ejercicio 2

•

Una extravagante repostería nos pidió ayuda para decorar su famosa torta cuadrada de chocolate:

	0	1	2	
2	1	1	1	2
1	1	1	1	1
0	1	1	1	0
	0	1	2	

La decoración consta de un confite de un mismo color en cada extremo de la torta. El color puede ser **Azul**, **Rojo**, **Verde** o **Negro**, ¡eso depende del gusto de quien encargue la torta! Si por ejemplo, alguien pide una torta con confites de color **Azul**, la torta decorada debería verse así:

	0	1	2	
2	1 1	1	1 1	2
1	1	1	1	1
0	1 1	1	1 1	0
	0	1	2	

Definí el procedimiento **DecorarTorta** que recibe un **color** como argumento y decora la torta con confites de ese color comenzando en el extremo Sur Oeste. No importa dónde termina el cabezal.

¡Dame una pista!

```
1 procedure DecorarTorta(color){
2   Poner(color)
3   IrAlBorde(Norte)
4   Poner(color)
5   IrAlBorde(Este)
6   Poner(color)
7   IrAlBorde(Sur)
8   Poner(color)
9 }
```



▶ Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:



Tablero inicial

	0	1	2	
2	1	1	1	2
1	1	1	1	1
0	1	1	1	0
	0	1	2	

Tablero final

	0	1	2	
2	1 1	1	1 1	2
1	1	1	1	1
0	1 1	1	1 1	0
	0	1	2	



Tablero inicial

	0	1	2	
2	1	1	1	2
1	1	1	1	1
0	1	1	1	0
	0	1	2	

Tablero final

	0	1	2	
2	1	1	1	2
1	1	1	1	1
0	1	1	1	0
	0	1	2	



Tablero inicial

	0	1	2	
2	1	1	1	2
1	1	1	1	1
0	1	1	1	0
	0	1	2	

Tablero final

	0	1	2	
2	1	1	1	2
1	1	1	1	1
0	1	1	1	0
	0	1	2	

Esta guía fue desarrollada por Mayra Mosqueira, Gustavo Trucco bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)





Te quedan 00:13:52

Ejercicio 3

Sabemos que no es saludable para nuestros oídos escuchar música a volúmenes muy altos 🗣️. Sin embargo, si está muy bajita tampoco escucharemos. Lo ideal es escucharla a un nivel entre 28 y 65. Para ello tenemos la función `estaEnRango` :

```
estaEnRango(40)
true // Porque está entre 28 y 65

estaEnRango(19)
false // Porque es menor que 28

estaEnRango(80)
false // Porque es mayor que 65
```

Definí la función `estaEnRango` que dado un volumen nos diga si está en el rango recomendable.

Solución >_ Consola

```
1 function estaEnRango(volumen){
2   return (volumen >= 28 && volumen <=65);
3 }
```

Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Mayra Mosqueira, Gustavo Trucco bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)





Te quedan 00:13:42

Ejercicio 4

Vamos a desarrollar un GPS que nos recomiende un destino a partir de una dirección . Para ello definiremos una función que reciba una dirección y dos destinos y según el valor del primer argumento nos recomiende hacia donde ir. Las únicas direcciones posibles son "norte" y "oeste". En caso que el primer argumento sea "norte" nos dirá que vayamos al primer destino, si es "oeste" nos recomendará que vayamos al segundo:

```
haciaDondeVamos("norte", "Gral. Las Heras", "Merlo")  
"Vamos a Gral. Las Heras"  
  
haciaDondeVamos("oeste", "Iguazú", "El Pato")  
"Vamos a El Pato"
```

Definí la función `haciaDondeVamos`.

¡Dame una pista!

Solución >_ Consola

```
1 function haciaDondeVamos(direccion, destino1, destino2){  
2   if (direccion === "norte"){  
3     return "Vamos a " + destino1  
4   } else { return "Vamos a " + destino2  
5 }}
```


 Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Mayra Mosqueira, Gustavo Trucco bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)





Te quedan 00:13:34

Ejercicio 5

Un local gastronómico quiere clasificar su vajilla y contar cuántos "vaso" s tiene a partir de una lista:

```
clasificarVajilla(["jarra", "vaso", "taza", "vaso", "vaso", "bowl"])
```

3

```
clasificarVajilla(["vaso", "taza", "taza", "bowl"])
```

1

Definí la función `clasificarVajilla` que a partir de una lista con la vajilla nos dice la cantidad de "vaso" s que tiene.

Solución Consola

```
1 function clasificarVajilla(vasos){
2   let sumatoria = 0;
3   for (let vaso of vasos){
4     if (vaso === "vaso"){
5       sumatoria += 1
6     }
7   }
8   return sumatoria;
9 }
10
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Mayra Mosqueira, Gustavo Trucco bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)





Te quedan 00:13:27

Ejercicio 6

En una casa de comidas guardan registro de los envíos que realizan a sus clientes 🍷. Estos registros tienen la siguiente forma:

```
let envioCalleFalsa = {
  direccion: "Calle Falsa 123",
  pedidos: ["Muzzarella", "Empanadas de verdura", "Papas fritas"],
  ultimoPedido: "15/11/2021"
}

let envioWallaby = {
  direccion: "Wallaby 42",
  pedidos: ["Ravioles con fileto", "10 piezas de sushi"],
  ultimoPedido: "16/12/2021"
}
```

Definí la función `resumenDeLosEnvios` que permita obtener un resumen de la información registrada de esta manera:

```
resumenDeLosEnvios(envioCalleFalsa)
"Calle Falsa 123 hizo su último pedido el 15/11/2021 y tiene registrados 3 pedidos"

resumenDeLosEnvios(envioWallaby)
"Wallaby 42 hizo su último pedido el 15/11/2021 y tiene registrados 2 pedidos"
```

Solución >_ Consola

```
1 function resumenDeLosEnvios(envio){
2   return envio.direccion + " hizo su último pedido el " +
  envio.ultimoPedido + " y tiene registrados " +
  (longitud(envio.pedidos)) + " pedidos"
3 }
4
```

 Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Mayra Mosqueira, Gustavo Trucco bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)





Te quedan 00:13:20

Ejercicio 7

¡Dejemos atrás a JavaScript para pasar a Ruby!

En esta ocasión queremos desarrollar parte de un juego, para ello vamos a modelar a su personaje principal: `Alvin`. Este personaje va a recolectar monedas y sabemos que:

- inicialmente tiene 6 monedas;
- puede duplicar sus monedas;
- si tiene más de 50 monedas diremos que es `profesional?`.

Definí en Ruby, el objeto `Alvin` que tenga un atributo `@monedas` con su getter. El objeto entiende los mensajes `duplicar!` (que multiplica por 2 su cantidad de monedas) y `profesional?`. No te olvides de inicializar el atributo `@monedas` con el valor correspondiente.

Solución **Consola**

```
1 module Alvin
2   @monedas = 6
3
4   def self.monedas
5     @monedas
6   end
7
8   def self.duplicar!
9     @monedas = @monedas * 2
10  end
11
12  def self.profesional?
13    @monedas > 50
14  end
15 end
```

Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Mayra Mosqueira, Gustavo Trucco bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)





Te quedan 00:13:12

Ejercicio 8

En un curso tenemos un conjunto de estudiantes, a la hora de cerrar las actas es necesario saber cuántas personas aprobaron . Teniendo en cuenta que cada estudiante sabe responder al mensaje `aprobo?` ...

Definí en Ruby el método `cantidad_de_personas_aprobadas` que responda a cuántas personas aprobaron de `Curso`.

Solución Consola

```
1 module Curso
2   @estudiantes = [May, Gus, Ro, Agus, Lu, Ale]
3   def self.cantidad_de_personas_aprobadas
4     @estudiantes.count { |estudiantes| estudiantes.aprobo? }
5   end
6 end
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Mayra Mosqueira, Gustavo Trucco bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2021  Mumuki

[Información importante](#)


[Términos y Condiciones](#)





 **Te quedan 00:13:04**

Ejercicio 9

A la hora de hacer turismo, es recomendable tener en cuenta qué lugares son interesantes para recorrerlos . Sabemos que:

- Los **Monumento** s son interesantes si tienen más de **150** años.
- Los **Museo** s siempre son interesantes.
- Los **Puente** s no son interesantes.

Definí el método `lugares_interesantes` en la clase `Lugar` que devuelva un listado de atracciones interesantes. Para eso deberás definir el método `interesante?` en los distintos tipos de atracciones.

 **Solución**  **Consola**

```
1 class Lugar
2   def initialize(unas_atracciones)
3     @atracciones = unas_atracciones
4   end
5   def lugares_interesantes
6     @atracciones.select{|atraccion| atraccion.interesante?}
7   end
8 end
9
10 class Monumento
11   def initialize(unos_anios)
12     @anios = unos_anios
13   end
14
15   def interesante?
16     @anios > 150
17   end
18 end
19
20 class Museo
```

```
21
22  def interesante?
23    true
24  end
25 end
26
27 class Puente
28   def interesante?
29     false
30   end
31 end
```

▶ Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Mayra Mosqueira, Gustavo Trucco bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2021  Mumuki

[Información importante](#)

[Términos y Condiciones](#)

