Examen #SeProgramar - Diciembre 2021 T1

Este es el examen de certificación de #SeProgramar. Lee bien cada consigna y tené en cuenta la siguiente información:

🕒 Revisá que la fecha y hora de tu dispositivo sean correctas.

⏳ Una vez que ingresás, tendrás 2 horas para resolver el examen, o hasta que termine el turno (lo que suceda primero).

⚠️ Si salís de la plataforma, el tiempo seguirá corriendo.

💾 Cada vez que enviás una solución, ésta se guarda en nuestro sistema.

✅ Una vez que terminás, simplemente podés salir de la plataforma: todas tus soluciones son almacenadas.

🚫 Cuando terminás no recibirás ninguna notificación automática.

✉️ Los resultados del examen serán enviados vía mail antes de la próxima semana.

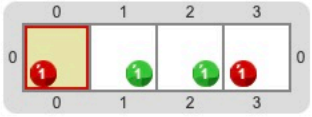
¡Te deseamos mucha suerte y éxitos! 🍀💪

Ejercicios

¡Comenzá esta lección!

Ejercicio 1: 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 `Rojo`, al `Este` una de color `Verde`, al `Este` una de color `Verde` y al `Este` una de color `Rojo`.

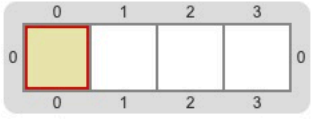
> Creá el programa que haga la caja de bombones solicitada. El cabezal comienza en el extremo Sur Oeste y no importa dónde termina.

```
program {
  Poner(Rojo)
  Mover(Este)
  Poner(Verde)
  Mover(Este)
  Poner(Verde)
  Mover(Este)
  Poner(Rojo)
}
```

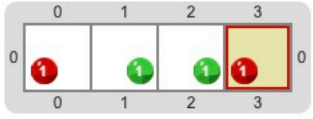
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



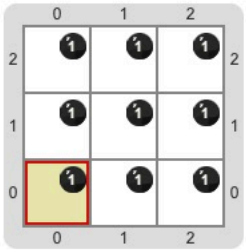
Tablero final



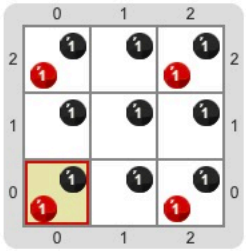
Siguiente Ejercicio: Ejercicio 2 >

Ejercicio 2: Ejercicio 2

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



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 `Rojo`, la torta decorada debería verse así:



> Definí el procedimiento `ColocarConfites` 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!

Tené en cuenta que el procedimiento solo debe poner los confites, la torta ya está creada. 😉

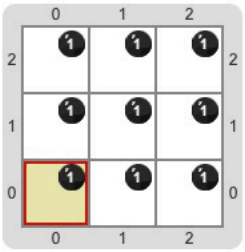
```
procedure ColocarConfites(color) {
  Poner(color)
  Mover(Norte)
  Mover(Norte)
  Poner(color)
  Mover(Este)
  Mover(Este)
  Poner(color)
  Mover(Sur)
  Mover(Sur)
  Poner(color)
}
```

Enviar

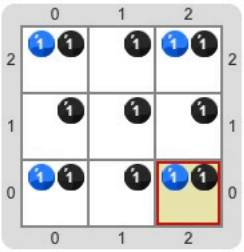
¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:

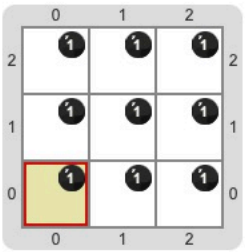
Tablero inicial



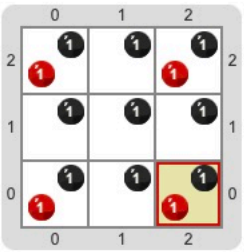
Tablero final



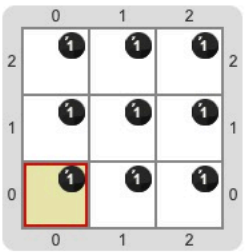
Tablero inicial



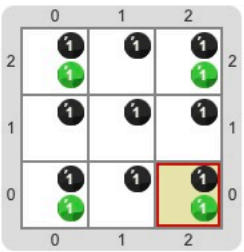
Tablero final



Tablero inicial



Tablero final



Siguiente Ejercicio: Ejercicio 3 >

Ejercicio 3: Ejercicio 3

JS

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 31 y 63. Para ello tenemos la función `esRecomendable`:

```
esRecomendable(40)
true // Porque está entre 31 y 63

esRecomendable(19)
false // Porque es menor que 31

esRecomendable(80)
false // Porque es mayor que 63
```

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

Solución | Consola

```
function esRecomendable(volumen) {
  return volumen >= 31 && volumen <= 63;
}
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Siguiente Ejercicio: Ejercicio 4 >

Ejercicio 4: Ejercicio 4

JS

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 `"noreste"` y `"sur"`. En caso que el primer argumento sea `"noreste"` nos dirá que vayamos al primer destino, si es `"sur"` nos recomendará que vayamos al segundo:

```
↳dondeVamos("noreste", "Gral. Las Heras", "Merlo")
"Vamos a Gral. Las Heras"

↳dondeVamos("sur", "Iguazú", "El Pato")
"Vamos a El Pato"
```

> Definí la función `dondeVamos`.

¡Dame una pista!

Tené en cuenta que la palabra `Vamos` en el string que retornamos está con la v en mayúscula. 😉

Solución | Consola

```
1 function dondeVamos(direccion, destino1, destino2) {
2   if (direccion === "noreste") {
3     return "Vamos a " + destino1;
4   } else {
5     return "Vamos a " + destino2;
6   }
7 }
```

Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

Siguiente Ejercicio: Ejercicio 5 >

Ejercicio 5: Ejercicio 5

JS

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

```
cuantosHay(["jarra", "recipiente", "taza", "recipiente", "recipiente", "bowl"])
3

cuantosHay(["recipiente", "taza", "taza", "bowl"])
1
```

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

Solución | Consola

```
function cuantosHay(lista) {
  let cantidad = 0;
  for (let vajilla of lista) {
    if(vajilla === "recipiente") {
      cantidad += 1;
    }
  }
  return cantidad;
}
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 6: Ejercicio 6

JS

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 `resumenDeInformacion` que permita obtener un resumen de la información registrada de esta manera:

```
resumenDeInformacion(envioCalleFalsa)
"Calle Falsa 123 tiene como fecha de último pedido el 15/11/2021 y tiene hechos en total 3 pedidos"

resumenDeInformacion(envioWallaby)
"Wallaby 42 tiene como fecha de último pedido el 15/11/2021 y tiene hechos en total 2 pedidos"
```

Solución | Consola

```
function resumenDeInformacion(registro) {
  return registro.direccion + " tiene como fecha de último pedido el " + registro.ultimoPedido + " y tiene hechos en total " + (longitud(registro.pedidos)) + " pedidos"
}
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Siguiente Ejercicio: Ejercicio 7 >

Ejercicio 7: 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: `Atrix`. Este personaje va a recolectar monedas y sabemos que:

- inicialmente tiene 4 monedas;
- puede duplicar sus monedas;
- si tiene más de 75 monedas diremos que es `increible?`.

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

Solución | Consola

```ruby
module Atrix
  @monedas = 4

  def self.monedas
    @monedas
  end

  def initialize(monedas)
    @monedas = monedas
  end

  def self.duplicar!
    @monedas *= 2
  end

  def self.increible?
    @monedas > 75
  end
end

```

Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

Siguiente Ejercicio: Ejercicio 8 >

Ejercicio 8: 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_materia?`...

> Definí en Ruby el método `cuantas_personas_aprobaron` que responda a cuántas personas aprobaron de `Estudiantado`.

Solución | Consola

```ruby
module Estudiantado
  @estudiantes = [May, Gus, Ro, Agus, Lu, Ale]

  def self.cuantas_personas_aprobaron
    @estudiantes.count { |estudiante| estudiante.aprobo_materia? }
  end
end
```

Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

Siguiente Ejercicio: Ejercicio 9 >

Ejercicio 9: Ejercicio 9

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

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

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

Solución | Consola

```ruby
class Pais
  def initialize(unas_atracciones)
    @atracciones = unas_atracciones
  end

  def atracciones_interesantes
    @atracciones.select {|unas_atracciones| unas_atracciones.interesante?}
  end
end

class Mausoleo
  def initialize(unos_anios)
    @anios = unos_anios
  end

  def interesante?
    @anios > 150
  end
end

class Museo
  def interesante?
    true
  end
end

class Puente
  def interesante?
    false
  end
end
```

Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Siguiente pendiente: Ejercicio 5 >