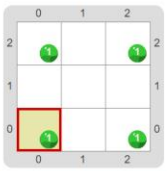
examen de argentina programa

Informática (Universidad Nacional de La Matanza)

Ejercicio 1: Decorando el balcón

Ale se aburrió de ver vacío su balcón. Así que compró 4 macetas con plantas para ponerlas en cada esquina, de esta forma:



Creá un programa que ponga una maceta (bolita `Verde`) en cada esquina del balcón. El cabezal empieza en el origen (o sea, en el borde Sur-Oeste) pero no te preocupes por dónde finaliza.

```
program {
  Poner(Verde)
  IrAlBorde(Norte)
  Poner(Verde)
  IrAlBorde(Este)
  Poner(Verde)
  IrAlBorde(Sur)
  Poner(Verde)
}
```

Enviar (998 intentos restantes)

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 2: Maceta multicolor

A Ale no le gustó como quedó el color verde en las macetas 😀. ¡Programemos algo para poder probar como quedaría con cualquier color!

Definí el procedimiento `PonerMacetas` para que ponga macetas del `color` que reciba por parámetro en cada esquina. No te preocupes por donde termina el cabezal.

```
procedure PonerMacetas(color) {
  Poner(color)
  IrAlBorde(Norte)
  Poner(color)
  IrAlBorde(Este)
  Poner(color)
  IrAlBorde(Sur)
  Poner(color)
}
```

Enviar (1000 intentos restantes)

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 3: ¿Quién es más chico?

Dejemos atrás los tableros y... ¡Pasemos a JavaScript! 💪

A veces la matemática puede ser un poco tediosa 😖. La buena noticia es que ahora podemos crear funciones que nos ayuden a resolver estos problemas. 🎉

Para eso vamos a crear una función que reciba 3 números y nos diga si la resta entre los 2 primeros es menor al tercero. Por ejemplo:

```
laRestaEsMenor(4, 2, 8)
true //Porque 4 menos 2 es 2 y es menor a 8

laRestaEsMenor(12, 3, 5)
false //Porque 12 menos 3 es 9 y es mayor a 5
```

Definí la función `laRestaEsMenor`.

Solución | Consola

```
function laRestaEsMenor(numero1, numero2, numero3) {
  return (numero1 - numero2) < numero3;
}
```

Enviar (1000 intentos restantes)

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 4: ¿Que me pongo?

JS

Si hay algo que a Ale le molesta es o pasar frío ❄️ o abrigarse de más 🔥. Pero lo que sí sabe, más allá de la temperatura, es de qué color vestirse ese día. Para eso, pensó en una función que recibe una temperatura y un color y responde qué ropa de ese color ponerse. Si la temperatura es 20 grados o más, se pone una remera de ese color. Sino, se pone un buzo de ese color:

```
vestirseSegun(20, "negra")
"Remera negra"

vestirseSegun(5, "verde")
"Campera verde"

vestirseSegun(30, "violeta")
"Remera violeta"
```

Definí la función `vestirseSegun`.

Solución >_Consola

```
function vestirseSegun(temperatura, color) {
  if(temperatura >= 20) {
    return "Remera " + color;
  } else {
    return "Campera " + color;
  }
}
```

Enviar (999 intentos restantes)

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 5: Corta, larga, corta, larga

JS

Ale está haciendo un trabajo de investigación y nos pidió ayuda 💪. Necesita poder sumar la cantidad de letras de las palabras cortas ➕. Una palabra se considera corta si tiene 6 o menos letras. Veamos un ejemplo:

```
sumaDeLetrasDePalabrasCortas(["hola", "murcielago", "caballo", "c
hoclo", "poco", "luz", "sol"])
20
```

Definí la función `sumaDeLetrasDePalabrasCortas`.

Solución >_Consola

```
function sumaDeLetrasDePalabrasCortas(lista) {
  let sumatoria = 0;

  for (let palabra of lista) {
    if(palabra.length <= 6) {
      sumatoria += palabra.length;
    }
  }

  return sumatoria;
}
```

Enviar (999 intentos restantes)

¡Muy bien! Tu solución pasó todas las pruebas

Ale estudia Historia y pensó en crear una función que le ayude a hacer resúmenes 📋. Para eso, consiguió registros de hechos históricos con la siguiente forma:

```
let independenciaArgentina = {
  suceso: "La declaración de la Independencia de Argentina",
  anio: 1816,
  ciudad: "San Miguel de Tucumán"
};

let declaracionDerechosHumanos = {
  hecho: "La declaración universal de los Derechos Humanos",
  anio: 1948,
  ciudad: "París"
};
```

La función deberá devolver un resumen de la información registrada de manera simple. Por ejemplo:

```
resumenHechoHistorico(independenciaArgentina)
"La Declaración de la Independencia de Argentina sucedió hace 184 añ
os en San Miguel de Tucumán"

resumenHechoHistorico(declaracionDerechosHumanos)
"La Declaración Universal de los Derechos Humanos sucedió hace 72 añ
os en París"
```

Definí la función `resumenHechoHistorico` que nos permita obtener la información requerida.

Solución >_Consola

```
function resumenHechoHistorico(registro) {

  return registro.suceso + " sucedió hace " + (2020 -
registro.anio) + " años en " + registro.ciudad;

}
```

Enviar (999 intentos restantes)

¡Muy bien! Tu solución pasó todas las pruebas

¡Dejemos atrás a JavaScript para pasar a Ruby!

Vamos a modelar `Comida`s para poder:

- agregarle cucharadas de sal;
- ver si está demasiado salada, es decir, si tiene más de 3 cucharadas de sal.

> Definí en Ruby, la clase `Comida` que tenga un atributo `@cucharadas_sal` con su getter. Las comidas entienden los mensajes `sumar_cucharadas!` (que recibe la cantidad a agregar por parámetro) y `esta_demasiado_salada?`. No te olvides de definir un `initialize` que reciba las cucharadas de sal iniciales como parámetro.

Solución >_Consola

```ruby
class Comida

  def initialize(cantidad_inicial)
    @cucharadas_sal = cantidad_inicial;
  end

  #Getters
  def cucharadas_sal
    @cucharadas_sal;
  end

  #Setters
  #Methods
  def sumar_cucharadas!(cantidad)
    @cucharadas_sal += cantidad;
  end

  def esta_demasiado_salada?
    @cucharadas_sal > 3;
  end

end
```

Enviar (999 intentos restantes)

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 8: ¿Tu nombre?

Los compilados son discos que tienen la característica de recopilar canciones que comparten alguna característica: artista, época, género. Algunas de ellas con mayor duración que otras.

Teniendo en cuenta que las canciones saben responder al mensaje `nombre`...

> Definí en Ruby el método `nombres_de_canciones` que responda el nombre de las canciones del `Compilado`.

Solución >_Consola

```ruby
module Compilado
  @canciones = [AmorAusente, Eco, Agujas, ElBalcon, GuitarrasDeCarton]

  #Method
  def self.nombres_de_canciones
     @canciones.map { |cancion| cancion.nombre }
  end

end
```

Enviar (998 intentos restantes)

¡Muy bien! Tu solución pasó todas las pruebas

A la hora de relajarse muchas `Persona`s juegan con su mascota. Los animales hacen distintas cosas cuando juegan:

- A los `Perro`s les da hambre;
- los `Gato`s incrementan en 2 su nivel de felicidad;
- las `Tortuga`s no hacen nada.

> Definí el método `jugar_con_mascota!` en la clase `Persona` y el método `jugar!` en los distintos tipos de animales. Definí los getters necesarios en cada una.

Solución >_Consola

```ruby
class Persona
  def initialize(mascota)
    @mascota = mascota
  end

  #Method
  def jugar_con_mascota!
    @mascota.jugar!;
  end
end

class Perro
  def initialize()
    @tiene_hambre = false;
  end

  #Getters
  def tiene_hambre
   @tiene_hambre;
  end

  #Methods
  def jugar!
   @tiene_hambre = true;
  end
end

class Gato
  def initialize(nivel_de_felicidad)
    @nivel_de_felicidad = nivel_de_felicidad;
  end

  #Getters
  def nivel_de_felicidad
   @nivel_de_felicidad;
  end

  #Method
  def jugar!
    @nivel_de_felicidad += 2;
  end
end

class Tortuga
  #Method
  def jugar!
  end
end
```