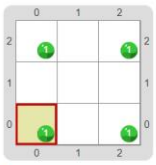
Examen 16 mayo 2022 14hs

argentina programa (Instituto Superior De Formacion Docente Jorge Luis Borges)

Ejercicio 1: Ejercicio 1

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

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 2: Ejercicio 2

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

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

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

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:

Ejercicio 3: Ejercicio 3

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

Cuando hacemos un examen de matemáticas es común verificar más de una vez si las cuentas que hicimos están bien.

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

```
> laMultiplicacionEsCorrecta(2, 4, 8)
true //Porque 2 por 4 es igual a 8

> laMultiplicacionEsCorrecta(3, 5, 12)
false //Porque 3 por 5 es 15, no 12
```

Definí la función `laMultiplicacionEsCorrecta`.

Solución Consola

```
function laMultiplicacionEsCorrecta(a,b,c){
  return a*b===c
}
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 4: Ejercicio 4

Ya pasamos por el tesoro de la matemática. Otro bien preciado es el tiempo . Es por ello que tratamos de usarlo sabiamente.

Con esto en mente vamos a crear una función que dados un nombre y un apodo nos diga cuál de los dos es más corto.

```
> conocerElMasCorto("Luis", "Lucho")
"Luis"

> conocerElMasCorto("Carolina", "Caro")
"Caro"

> conocerElMasCorto("Ricardo", "Ringo")
"Ringo"
```

Definí la función `conocerElMasCorto`.

Solución | Consola

```
function conocerElMasCorto(nombre,apodo){
  if (longitud(nombre)<longitud(apodo))
  {return nombre}
  else
  {return apodo}
}
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 5: Ejercicio 5

Vale está haciendo un trabajo de investigación y nos pidió ayuda . Necesita poder distinguir las palabras más cortas de una oración . Una palabra se considera corta si tiene menos de 5 letras. Veamos un ejemplo:

```
> filtrarCortas(["Hari", "Seldon", "nacido", "en", "el", "año", "11988", "de",
"la", "Era", "Galáctica"])

["Hari", "en", "el", "año", "de", "la", "Era"]
```

Definí la función `filtrarCortas`.

Solución | Consola

```
function filtrarCortas(lista){
  let otraLista=[]
  for(let palabra of lista) {
    if (longitud(palabra)<5){
      agregar(otraLista,palabra)
    }
  }
  return otraLista
}
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 6: Ejercicio 6

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 = {
  suceso: "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:

```
> resumenHito(independenciaArgentina)
"La Declaración de la independencia de Argentina ha sucedido hace 205 años en
la ciudad de San Miguel de Tucumán"

> resumenHito(declaracionDerechosHumanos)
"La Declaración Universal de los Derechos Humanos ha sucedido hace 73 años en
la ciudad de París"
```

Definí la función `resumenHito` que nos permita obtener la información requerida. Asumí que estamos en 2021.

Solución | Consola

```
function resumenHito(hecho){
  return hecho.suceso + " ha sucedido hace " + (2021-
hecho.anio) + " años en la ciudad de " + hecho.ciudad

      }
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 7: Ejercicio 7

¡Dejemos atrás a JavaScript para pasar a Ruby!

Vamos a modelar la clase `Tablet` para poder:

- cargar una cantidad de minutos determinada (si cargamos 10 minutos, su bateria incrementa en 10);
- ver si tiene carga suficiente, es decir, si su bateria es mayor a 55.

> Definí en Ruby, la clase `Tablet` que tenga un atributo `@bateria` con su getter. Las instancias de la clase `Tablet` entienden los mensajes `cargar!` (que recibe los minutos por parámetro y lo carga esa cantidad) y `carga_suficiente?`. No te olvides de definir un `initialize` que reciba a la bateria inicial como parámetro.

Solución | Consola

```ruby
class Tablet

  def bateria
    @bateria
  end

  def initialize(bateria)
    @bateria=bateria
  end

  def cargar!(min)
    @bateria += min
  end

  def carga_suficiente?
    @bateria > 55
  end

end
```

▶ Enviar

✔ ¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 8: Ejercicio 8

A `Oli` le gusta mucho leer , pero le interesa saber cuántos de los libros que leyó son largos. Cada uno de los libros sabe responder al mensaje `demasiado_largo?`.

> Definí en Ruby el método `cantidad_largos` que responda a cuántos libros largos leyó `Oli`.

Solución | Consola

```ruby
module Oli
  @libros_leidos = [MartinFierro, Fundacion, ElPrincipito, HarryPotter]

  def self.cantidad_largos
  @libros_leidos.count { |libro| libro.demasiado_largo?}
  end

end
```

▶ Enviar

✔ ¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 9: Ejercicio 9

Una productora de cine se encarga de invertir en las películas que trabajan con ella. Cuando esta empresa invierte:

- Las películas de tipo `Drama` duplican su `duracion`;
- las de `Suspenso` contratan 10 `extras`;
- las de género `Comedia` no se modifican.

> Definí el método `invertir_en_peliculas!` en la clase `Productora` y el método `recibir_inversion!` en los distintos tipos de películas. Definí los getters necesarios en cada una.

Solución | Consola

```ruby
class Productora
  def initialize(peliculas)
    @peliculas = peliculas
  end

  def invertir_en_peliculas!
    @peliculas.each {|pelicula| pelicula.recibir_inversion!}
  end

end

class Drama
  def duracion
    @duracion
  end
  def initialize(duracion)
    @duracion = duracion
  end

  def recibir_inversion!
    @duracion *= 2
  end

end

class Suspenso
  def extras
    @extras
  end
  def initialize(extras)
    @extras = extras
  end

  def recibir_inversion!
    @extras += 10
  end

end

class Comedia
  def recibir_inversion!
  end
end
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas