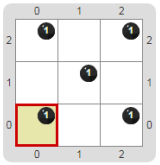
mumuki.io/argentina-programa/exercises/13253-examen-seprogramar-diciembre-2021-t10-ejercicio-1

Te quedan 01:57:41

Ejercicio 1: Ejercicio 1

Es bastante sabido que para recordar dónde se esconde un tesoro hay que marcar el lugar. □

Una clásica opción para esto es utilizar una cruz ✕, que en un tablero podría verse así:



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

```
program{
  Poner(Negro)
  Mover(Norte)
  Mover(Norte)
  Poner(Negro)
  Mover(Este)
  Mover(Este)
  Poner(Negro)
  Mover(Sur)
  Mover(Sur)
  Poner(Negro)
  Mover(Oeste)
  Mover(Norte)
  Poner(Negro)
}
```

Enviar

mumuki.io/argentina-programa/exercises/13254-examen-seprogramar-diciembre-2021-t10-ejercicio-2

Te quedan 01:56:24

Ejercicio 2: Ejercicio 2

La verdad es que en el ejercicio anterior hicimos una cruz de un color específico porque es lo que solemos ver en películas o libros pero ¿qué nos impide que hagamos una cruz de cualquier color para marcar un lugar? □

Definí el procedimiento `DibujarCruz` para que dibuje una cruz con el `color` que reciba por parámetro. No te preocupes por donde termina el cabezal.

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

Enviar

mumuki.io/argentina-programa/exercises/13255-examen-seprogramar-diciembre-2021-t10-ejercicio-3

Te quedan 01:54:04

Ejercicio 3: Ejercicio 3

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 mayor al tercero. Por ejemplo:

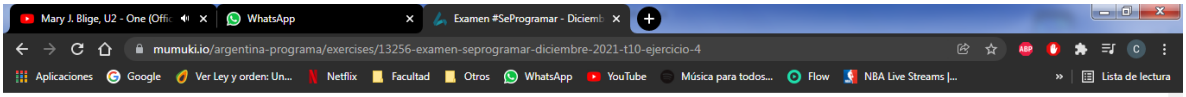
```
laRestaEsMayor(4, 2, 8)
false //Porque 4 menos 2 es 2 y es menor a 8

laRestaEsMayor(12, 3, 5)
true //Porque 12 menos 3 es 9 y es mayor a 5
```

Definí la función `laRestaEsMayor`.

Solución | Consola

```
function laRestaEsMayor(num1,num2,num3){
return((num1-num2)>num3)
}
```

Enviar



Te quedan 01:52:37

Ejercicio 4: Ejercicio 4

Ahora vamos a hacer una función un poco particular. □

Queremos crear un mezclador de palabras que reciba 2 palabras y un número. Si el número es menor o igual a 10 el mezclador concatena la primera palabra con la segunda. En cambio, si el número es mayor a 10, concantena la segunda con la primera:

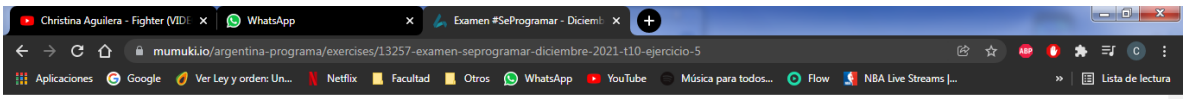
```
mezcladorDePalabras("planta", "naranja", 10)
"plantanaranja"

mezcladorDePalabras("amor", "amarillo", 9)
"amoramarillo"

mezcladorDePalabras("mate", "pato", 11)
"patomate"
```

Definí la función `mezcladorDePalabras`.

Solución | Consola

```
function mezcladorDePalabras(p1,p2,n){
  if(n<=10){
    return(p1+p2);
  }
  else{
    return(p2+p1);
  }
}
```

Enviar



Te quedan 01:50:27

Ejercicio 5: Ejercicio 5

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:

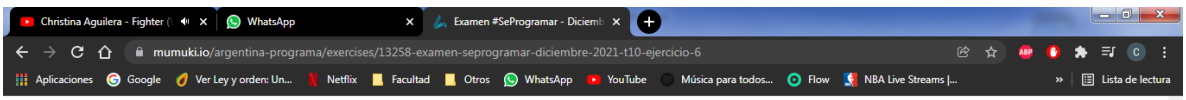
```
sumatoriaLetrasDePalabrasCortas(["hola", "murcielago", "caballo", "choclo", "poco", "luz", "sol"])
20
```

Definí la función `sumatoriaLetrasDePalabrasCortas`.

Solución | Consola

```
function sumatoriaLetrasDePalabrasCortas(palabras){
  let suma = 0;
  for(let palabra of palabras){
    if(longitud(palabra)<=6){
      suma = suma + longitud(palabra);
    }
  }
return(suma);
}
```

Enviar



Te quedan 01:47:01

Ejercicio 6: Ejercicio 6

Los servicios de películas bajo demanda lograron despertar un interés renovado en la sociedad por el cine y las series . Es por ello que contamos registros de este estilo:

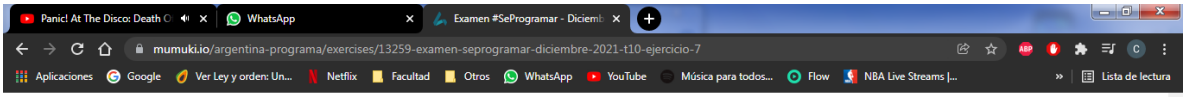
```
let gus = {
  nick: "Wuisti",
  promedioPeliculasMensuales: 5,
  plataforma: "NetFix"
};

let ariel = {
  nick: "Ari",
  promedioPeliculasMensuales: 10,
  plataforma: "Armazon"
};
```

Ahora debemos definir una función que permita obtener un resumen de la información registrada de manera simple. Por ejemplo:

Solución | Consola

```
function informacionResumida(persona){
  return( "Se estima que " + persona.nick + " verá " + (persona.promedioPeliculasMensuales*12) + " películas en un año con la plataforma " + persona.plataforma);
}
```

Enviar



Te quedan 01:44:55

Ejercicio 7: Ejercicio 7

¡Dejemos atrás a JavaScript para pasar a Ruby!

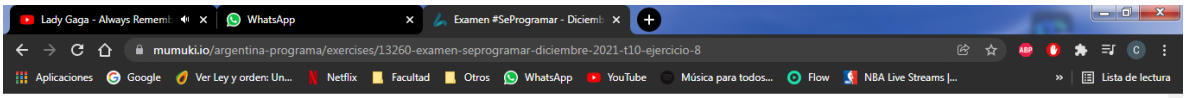
Vamos a modelar `Auto`s para poder:

- cargarle una cantidad de nafta determinada;
- ver si tiene carga suficiente, es decir, si tiene más de 48 litros de nafta.

> Definí en Ruby, la clase `Auto` que tenga un atributo `@nafta` con su getter. Los autos entienden los mensajes `cargar_nafta!` (que recibe la cantidad a cargar por parámetro) y `suficiente_combustible?`. No te olvides de definir un `initialize` que reciba a la nafta inicial como parámetro.

Solución | Consola

```ruby
class Auto

def initialize (nafta)
    @nafta = nafta
end

def nafta
    @nafta
end

def cargar_nafta!(cantidad)
@nafta += cantidad
end

def suficiente_combustible?
@nafta>48
end

end
```



Te quedan 01:41:48

Ejercicio 8: Ejercicio 8

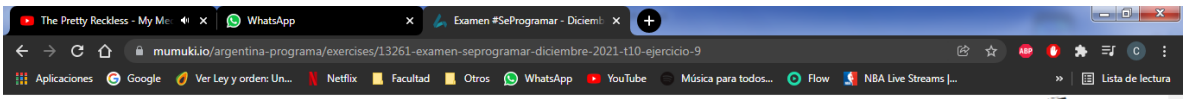
Los compilados son discos que tienen la característica de recopilar canciones que comparten alguna característica, por ejemplo artista, época o género. Algunas de ellas con mayor duración que otras.

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

> 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]

  def self.nombres_de_canciones
    @canciones.map {|cancion|cancion.titulo}
  end

end
```



Ejercicio 9: Ejercicio 9

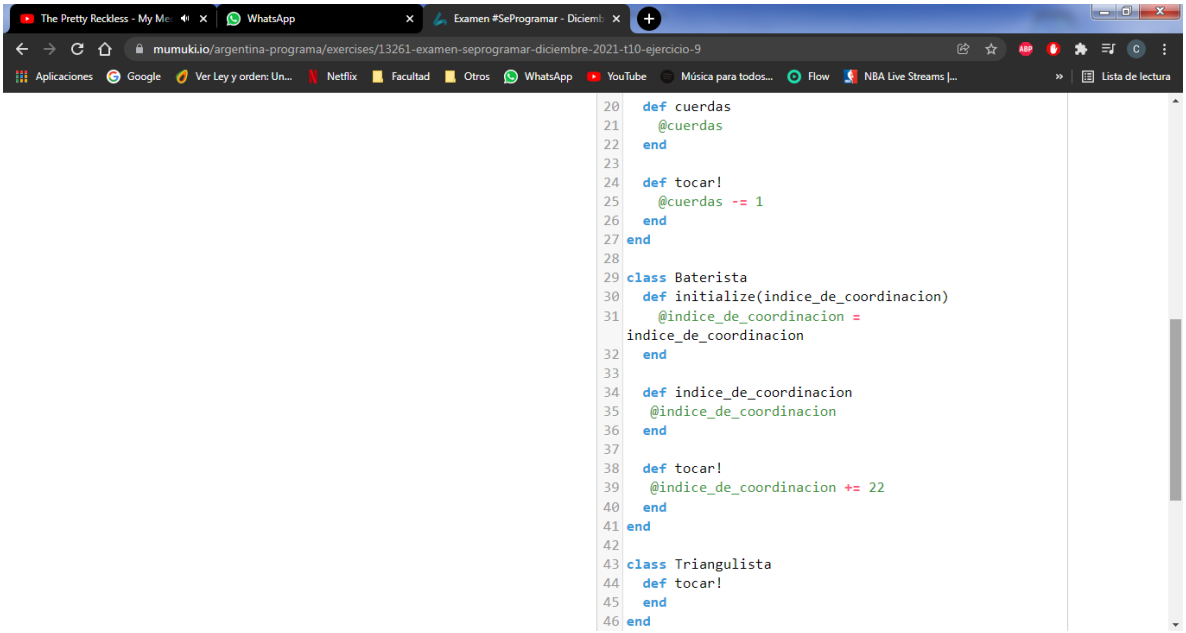
Como bien sabemos, una `Banda` tiene integrantes. Cuando la banda toca, toca cada integrante:

- `Violinista` pierde una de sus `cuerdas`;
- `Baterista` sube su `indice_de_coordinacion` en 22;
- `Triangulista` no hace nada.

> Definí el método `tocar!` tanto en la `Banda` como en los distintos tipos de integrantes. Definí los getters necesarios en cada integrante.

Solución | Consola

```ruby
class Banda
  def initialize(integrantes)
    @integrantes = integrantes
  end

  def integrantes
    @integrantes
  end

  def tocar!
   @integrantes.each{|integrante|integrante.tocar!}
  end
end

class Violinista
  def initialize(cuerdas)
    @cuerdas = cuerdas
  end

  def cuerdas
    @cuerdas
```




```ruby
  def cuerdas
    @cuerdas
  end

  def tocar!
    @cuerdas -= 1
  end
end

class Baterista
  def initialize(indice_de_coordinacion)
    @indice_de_coordinacion =
indice_de_coordinacion
  end

  def indice_de_coordinacion
   @indice_de_coordinacion
  end

  def tocar!
   @indice_de_coordinacion += 22
  end
end

class Triangulista
  def tocar!
  end
end
```