Ejercicios repaso tema 3

# **Función para calcular el volumen de un cono y testeo con Jasmine (volumen\_cono.html)**

Crea una función llamada **volumenCono** que calcule el volumen de un cono a partir del radio y de la altura. Utiliza posteriormente esta función mediante un programa que solicite al usuario el radio y la altura y posteriormente le muestre el volumen del cono.

Sintaxis:

**volumenCono (radio,altura)** ⇒ Devuelve el volumen del cono

(Debes hacer los cálculos usando las posibilidades que te ofrece la clase Math, como operaciones matemáticas y constantes)

El volumen de un cono viene dado por la fórmula de V indicada en la imagen:

|  | AL: Área lateral  AT: Área lateral  V: Volumen |
| --- | --- |

Puedes usar el siguiente enlace para comprobar que tu función calcula correctamente el volumen:

<http://www.calcularelvolumen.com/cono.htm>

Prepara un testeo con Jasmine con la información de la siguiente tabla:

| Entrada a la función | | Salida de la función |
| --- | --- | --- |
| radio | altura | Volumen  (redondeado a dos decimales) |
| 3 | 7 | 65.97 |
| 5 | 12 | 314.16 |
| 8.5 | 19.2 | 1452.67 |
| 4.5 | 14.1 | 299.00 |

# **Testeo con Jasmine de Jarra**

Prepara un **testeo con Jasmine** para la clase Jarra con las siguientes especificaciones:

* Crear una jarra1 de 10 litros de capacidad y que contenga 4 litros
* Crear una jarra2 de 15 litros de capacidad y que contenga 8 litros
* Invocar al métodos estático Jarra.comparar con jarra1 y jarra2 y comprobar que devuelve jarra2, ya que contiene más litros
* Invocar a jarra1.llenarDesde(jarra2) y comprobar que jarra1 contiene 10 libros y jarra2 2 litros
* Invocar al método vaciar() de jarra2 y comprobar que la cantidad de dicha jarra resulta ser de 0 litros

# **Función que devuelve elementos de un array sin duplicidades y ordenados (array\_sin\_duplicidad.html)**

Crea una función llamada **arraySinDuplicidad** a la cual le pases como argumento de entrada un array. La función debe devolver un array que contenga los elementos del array sin repetir y ordenados.

Sintaxis:

***arrayNoRepetido(array)*** => devuelve array con eltos del array de entrada sin duplicidades y ordenados

Prueba tu función pasándole el array:

[4,0,4,7,7,5,8,1,8,8,0,2,3,1,2,5,7,3,2,5,1]

Tu función debería devolver el array:

[0,1,2,3,4,5,7,8]

# **Función que busca las posiciones de un determinado elemento en un array (buscar\_posiciones\_array.html)**

Crea una función llamada **buscaPosiciones** a la cual le pases dos argumentos, el primer argumento será un array de números y segundo un número a buscar. La función debe devolver un array con todas las posiciones donde ha encontrado el número a buscar en el array. Tu función debe utilizar internamente el método indexOf sobre el array de búsqueda.

Sintaxis:

**buscarPosiciones**(array, elto) ⇒ Devuelve array con las posiciones del array de entrada donde se ha encontrado el segundo argumento.

Por ejemplo si se invoca a la función con los siguiente argumentos:

buscarPosiciones ([4,0,4,7,5,8,1,8,0,2,3,1],1)

La función debe devolver el array:

[6,11]

(Ya que el elemento 1 se encuentra dentro del array en las posiciones 6 y 11)

# **Función que devuelve el número de palabras de un texto (contar\_palabras.html)**

Crea una función llamada **contarPalabras** que tenga como argumento de entrada una cadena de caracteres y devuelva el número de palabras existentes en la cadena.

Prueba tu función pasándole la frase:

"por cien cañones por banda poema de Espronceda"

Tu función debería devolver:

8

# **Función que genera una bonoloto y testeo con Jasmine (bonoloto.html)**

**Apartado a)**

Crea una función llamada **bonoloto** que devuelva un array con 6 números aleatorios entre el 1 y el 49. Los números aleatorios no pueden repetirse, ya que estamos simulando una bonoloto. Haz también que el array devuelto esté ordenado.

Si invocamos a la función bonoloto(), una posible salida de la función podría ser:

# **[6, 10, 16, 22, 39, 42]**

**Apartado b)**

Crea una función llamada **compruebaDuplicidadArray** que le pasemos como argumento de entrada un array de números y devuelva true si tiene algún elemento repetido (o false en caso contrario). Testea tu función usando un array de valores de entrada (serán arrays de números) y valores esperados (true/false) dependiendo de si tienen valores repetidos o no.

**Apartado c)**

Crea una función llamada **compruebaEltosArrayEnRango** que le pasemos como argumento de entrada un array de números y devuelva true si todos los elementos del array están comprendidos en el rango de 1 a 49.

**Apartado d)**

**Testea** la función creada en el apartado A (bonoloto). Genera 100.000 bonolotos comprobando que ninguna de ellas tengan números repetidos y que todos los elementos están en el rango de 1 a 49.

# 

|  |
| --- |