**Banco de ejercicios de JavaScript I**

1. Crea una función llamada average que tome por parámetro un array llamado dataArray y calcule la media de los valores almacenados en el array. La función deberá devolver la media de dicho vector y undefined en caso de que el array no tenga elementos. Emplea funciones de orden superior sobre arrays (e.g., map, filter, reduce, etc.)
2. Crea una función llamada linearSearch que tome un parámetro x de tipo numérico, y un array llamado dataArray. La función deberá devolver cierto en caso de que el valor x se encuentre en dataArray, y falso en caso contrario. No se puede emplear el método/instrucción include definido para los arrays. Emplea la nueva sintaxis de bucles for.
3. Crea una función llamada getMinimum que tomo un array numérico llamado dataArray como parámetro, y devuelva el valor mínimo encontrado el dicho array. Emplea la función de orden superior sobre arrays reduce.
4. Crea una función llamada getMaximum que tomo un array numérico llamado dataArray como parámetro, y devuelva el valor mínimo encontrado el dicho array. Emplea la función de orden superior sobre arrays reduce.
5. Crea una función que tome como parámetro un array de números llamado dataArray, y que devuelva un nuevo array con el resultado de elevar todos los elementos al cuadrado, únicamente manteniendo aquellos cuyo cuadrado es par. Emplea funciones de orden superior (e.g., map, filter, reduce, etc.)
6. Crea una función llamada everyElementGreaterThan que tome por parámetro un número x, y un array de datos dataArray. La función devolverá cierto en caso de que TODOS los elementos sean mayores que x, y falso en caso contrario (e.g., map, filter, reduce, etc.)
7. Crea una función llamada getClosestEnemyIndex que tome como parámetro un array de dos posiciones representando las posiciones x e y de un personaje llamado characterPosition, y un segundo parámetro que será un array de enemigos llamado enemyPositions donde cada valor introducido será un array con las posiciones x e y de un enemigo. La función deberá determinar el índice en el array del enemigo más cercano al personaje. En principio, emplea la distancia euclidea para determinar la distancia entre dos pares de coordenadas (<https://en.wikipedia.org/wiki/Euclidean_distance#Two_dimensions>). Puedes construirte una función llamada euclideanDistance que tome como parámetros dos coordenadas, y calcule la distancia euclidea entre ambas posiciones. Emplea para este ejercicio la función de orden superior reduce.
8. Crea una función llamada multipleFactorial que tome como parámetro un array de número llamado dataArray, y devuelva un núevo array que sea el resultado de calcular el factorial para cada uno de los elementos en el array. Emplea funciones de orden superior (e.g., map, filter, reduce, etc.)
9. Crea una función que tome un array de números llamado dataArray como parámetro y determine si algún elemento del array es múltiplo de 15.
10. Crea una función que tome un array de nombres de usuario llamado usersArray, y un array de nombres de usuarios baneados llamado blackList, y que devuelva un nuevo array con los usuarios no baneados en el array inicial. Emplea funciones de orden superior (e.g., filter, map, reduce)
11. Crea una función que tome por parámetro una cadena de texto y retorne en un array todos aquellos elementos que han aparecido al menos tres veces en la cadena de texto. Emplea la estructura de datos map.
12. Crea una función que simule el lanzamiento de un dado de 6 caras un número determinado de veces recogido en un parámetro llamado n. La función deberá devolver cuantas veces ha aparecido cada una de las caras del dado en los experimentos. Emplea la estructura de datos map.
13. Crea un programa que permita al usuario crear una lista de la compra. El programa solicitará ítems al usuario hasta que éste introduzca el carácter especial \*. La lista de la compra no permitirá duplicados. Al final del programa se imprimirán por el terminal todos los elementos de la lista de la compra. Emplea la estructura de datos map o set.
14. Crea una función que tome por parámetro un texto llamado text y que determine cual es el carácter más común en dicho texto. El carácter será retornado por la función.
15. Crea una clase para representar corredores de maratón dentro de un programa. Los corredores tienen un nombre, un apellido, una edad, un club, y una categoría (primera, segunda, o tercera). En el momento de creación del corredor, puede que tan solo conozcamos el nombre y el apellido. El corredor también tiene marcas realizadas durante la temporada. Crea la estructura de clase adecuadas, y además implementa los siguientes métodos:
    * Añadir marca, que permita añadir una marca al corredor
    * Obtener la marca mas baja, que permita obtener la marca más baja del corredor en la temporada
    * Obtener la marca más alta, que permita obtener la marca más alta del corredor en la temporada
    * Un método que nos diga si el corredor es menor de edad