Aplicaciones Telemáticas, examen práctico, 15/05/2026
Grado en Ingeniería Telemática
Universidad Rey Juan Carlos

Instrucciones

- Emplea lo visto en clase, y solo lo visto en clase (que puedes consultar en las transparencias). No será válido ni JavaScript más antiguo ni más avanzado. Usa el modo estricto.

Ejercicio 1. (4.0 puntos)

Ya conoces la *paradoja del cumpleaños*. Definiremos ahora una idea similar de la siguiente manera: si dos personas en un grupo cumplen años en la misma semana (según la definición que veremos a continuación), consideramos que ya se cumple la *paradoja laxa*. Escribe un programa en JavaScript llamado *laxa.js* según la siguiente especificación.

1. Su función principal se llamará *laxa()*. Recibirá dos argumentos, el primero será un array como este:

```
[
    "3 enero,8 enero,12 enero",
    "1 febrero,15 abril,30 julio",
    "27 diciembre,2 enero,14 mayo,1 septiembre",
    "5 junio,5 junio,5 junio"
]
```

   Esto es:

   a) Un array de cadenas. El array podrá estar vacío.

   b) Cada cadena tendrá al menos una fecha, no podrá ser la cadena vacía.

   c) Entre fecha y fecha, habrá una coma. Tras la última fecha, no.

   d) Cada fecha estará compuesta por el día del mes, un espacio y el nombre del mes.

   e) El mes estará escrito en español, todo con minúsculas.

   f) Ignoraremos los años bisiestos.

   g) No debes preocuparte por lo que haga el programa si las cadenas incumplen este formato. Suponemos que los datos ya están previamente filtrados, y si el filtro es incorrecto, no es responsabilidad de esta función. Ejemplos de fechas erróneas que puedes ignorar:

```
""                  Fecha vacía.
"21abril"           Falta el espacio
"30 febrero"        Es una fecha errónea.
"21  abril"         Sobra un espacio entre la fecha y el mes.
"21 abril "         Sobra un espacio al final.
"21 Abril"          Hay una mayúscula.
"21 april"          No está en español.
"29 febrero"        Es válido solo en bisiesto.
... entre otros casos.
```

2. El segundo argumento de la función *laxa()* será un booleano. Si vale *true*, el programa irá escribiendo con *console.log()* una traza describiendo lo que hace: día del año de cada fecha, diferencia entre fechas, etc. El detalle de esta traza queda a tu criterio.

3. La definición que aplicaremos aquí de *misma semana* es la siguiente: dadas dos fechas, calculamos su día del año (entre 1 y 365, o entre 0 y 364). Calcularemos la diferencia en valor absoluto entre ambos valores y si es 6 o menos, consideramos que se trata de la misma semana. Para que no haya inconsistencias entre fin de un año y comienzo del siguiente, también consideraremos que se trata de la misma semana si la diferencia en valor absoluto es 359 o más (por ejemplo entre el 30 de diciembre y el 2 de enero la diferencia sería 362). Puedes reutilizar código de tu práctica 3.12.

4. Atención: si por ejemplo hay un 7 de mayo y un 10 de mayo, esto es 1 coincidencia. No vuelvas a contar el 10 de mayo y el 7 de mayo como una coincidencia adicional.

5. Cada cadena representa un grupo independiente de personas. Solo son relevantes las coincidencias dentro de la misma cadena, no entre cadenas distintas.

6. Si una misma fecha aparece varias veces, cada aparición se considera una persona distinta. Ejemplo: `"5 junio,5 junio,5 junio"` son tres personas y tres coincidencias: el primero con el segundo, el primero con el tercero y el segundo con el tercero.

7. La función devolverá un array. Cada elemento del array contendrá el número de pares de personas que cumplen esta *paradoja laxa* para cada una de las cadenas de entrada.

   Para el ejemplo mostrado anteriormente, el resultado sería `[2,0,1,3]`

8. Si la función no recibe exactamente dos argumentos, lanzará una excepción, de forma similar a como lo hace el motor de JavaScript.

   También lo hará si el primer argumento no es un array, si alguno de los elementos del array no es una cadena o si el segundo argumento no es un booleano. Aunque, como hemos dicho, no debes comprobar si las cadenas son correctas.

9. En el fichero *laxa.js*, escribe una invocación a *laxa()*, pasándole el ejemplo del enunciado. Lo encontrarás en el aula virtual.

Ejercicio 2. (6.0 puntos)

Modifica tus ficheros *carta.html* y *carta.js* para que:

- El sistema gestione internamente el stock (nº de unidades disponibles de cada producto). En cada ejecución, el programa asignará a cada producto un número aleatorio de unidades. De la forma que prefieras, pero que no sea un número muy alto, para poder probarlo cómodamente.
- Cada vez que un usuario añada una unidad al pedido, se decrementará del stock. Observa que no es necesario que el cliente finalice el pedido, basta con que lo añada a su pedido.
- Si no quedan productos, esto se mostrará claramente en la carta. De la manera que veas conveniente. El cliente no podrá pedir productos sin stock. Si lo intenta, el comportamiento de la aplicación queda a tu criterio, pero debe ser *razonable*.
- Si el cliente cancela pedidos, las unidades disponibles se actualizarán. Naturalmente, esto podrá provocar que productos agotados ya no lo estén. El programa actuará en consecuencia.
- Normalmente los clientes no deberían ver cuánto producto queda, solo si puede pedir algo o no. Para ello, el programa tendrá dos modos de funcionamiento.
  - *Modo cliente*. La carta no muestra el stock.
  - *Modo administrador*. La carta sí muestra el stock.

  Un botón en la carta permitirá pasar de un modo a otro. La aplicación responderá inmediatamente a este botón. Cuando el programa esté en *modo cliente*, el texto del botón será *modo administrador*. Y viceversa. Esto es, el botón indica qué hará el botón si se pulsa, no el modo actual.