Instrucciones:

- *Resuelve cada uno de los problemas* ***en hojas separadas****.*
- *En la resolución de los problemas* ***no puedes utilizar*** *elementos del lenguaje Python no estudiados en la asignatura, como son las tuplas, los diccionarios, las listas por comprensión, las estructuras* `for-else`*, los métodos* `index` *o* `count`*, las funciones* `sum`*,* `max`*,* `min`*,* `mean`*, etc.*

Problema 1 (5 puntos)

A partir de los datos de una agencia meteorológica, hemos creado una matriz de C filas por D columnas en las que en cada fila guardamos la temperatura media observada cada día de un periodo de D días, en cada una de C ciudades europeas. El nombre de cada ciudad se almacena en una lista de tal forma que la posición `i` de la lista se corresponde con la fila `i` de la matriz. Ten en cuenta que las ciudades y los días se numeran empezando por cero.

Por ejemplo, la lista de ciudades y matriz de temperaturas podrían ser las siguientes:

```
ciudades = ['Reykjavik', 'Rovaniemi', 'Valetta', ...]

temperaturas = [[ -3,  -2,  -2,   1,   4,   7,   9,   8,   6,  -2,  -1,  -2, ...],  # Reykjavik
                [-16, -14, -10,  -3,   3,   8,  12,   9,   5,  -1,  -6, -11, ...],  # Rovaniemi
                [  9,   9,  10,  12,  15,  19,  21,  22,  20,  17,  14,  11, ...],  # Valetta
                 ...
               ]
```

Implementa en Python:

a) (2,5 PUNTOS) Una función, `ciudades_con_ola_de_frío`, que tenga como parámetros una matriz de temperaturas, su lista de nombres de ciudades correspondiente y un valor entero n. La función debe devolver una lista con los nombres de las ciudades que han sufrido alguna ola de frío. Se considera que una ola de frío se produce cuando se observa una temperatura bajo cero durante al menos n días consecutivos.

b) (2,5 PUNTOS) Un procedimiento, `mostrar_ciudades_extremas`, que tenga como parámetros una matriz de temperaturas, su lista de nombres de ciudades correspondiente y dos números enteros que representan el día de inicio y de fin de un rango de días. El procedimiento debe mostrar por pantalla la ciudad y temperatura más alta y más baja de cada día en el rango dado, incluyendo el día inicial y el final. En caso de empate entre dos o más ciudades, se puede elegir cualquiera de ellas. Si el rango es incorrecto, el procedimiento debe mostrar un mensaje de error. Si consideramos, por ejemplo, el rango entre el día 8 y el día 10, la salida debería tener un formato similar al mostrado a continuación:

```
Día 8: mínima de 5 en Rovaniemi y máxima de 20 en Valetta.
Día 9: mínima de -2 en Reykjavik y máxima de 17 en Valetta.
Día 10: mínima de -6 en Rovaniemi y máxima de 14 en Valetta.
```

Problema 2 (5 puntos)

Queremos implementar una aplicación para gestionar las valoraciones de los usuarios de todos los establecimientos hosteleros de España. Para ello, disponemos de una serie de ficheros de texto con las valoraciones recogidas en diferentes páginas web. Cada línea de uno de estos ficheros corresponde a una valoración y contiene, separados por el carácter `#`, el identificador del usuario que realiza la valoración, el nombre del establecimiento, la localidad donde está ubicado y un número entero que representa la valoración (de 0 a 5 estrellas) asignada por ese usuario. A continuación se muestran, a modo de ejemplo, algunas líneas de un fichero como el descrito:

```
user12345678#Hostal Jacinta#Sagunto#4
user33468648#Gran Hotel Don Pedro#Almazora#1
user62213441#Hotel París#León#5
user62213441#Hostal Jacinta#Sagunto#3
user28821345#Hotel París#León#5
```

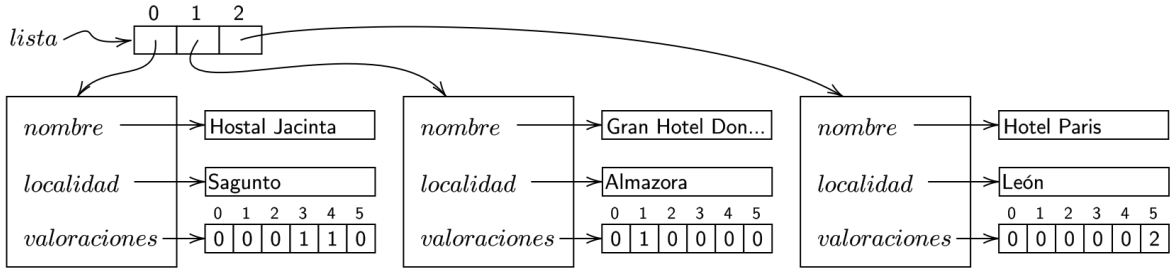
Considera la clase `Establecimiento`, definida como sigue:

```
class Establecimiento:
    def __init__(self, nombre, localidad):
        self.nombre = nombre
        self.localidad = localidad
        self.valoraciones = [0] * 6  # Seis contadores para almacenar el total de veces
                                     # que ha recibido cada valoración de 0 a 5 estrellas
    def añadir(self, estrellas):
        self.valoraciones[estrellas] += 1
```

Como puedes ver, los objetos de la clase `Establecimiento` almacenan el nombre y la localidad de un establecimiento y una lista `valoraciones` con seis contadores. Así, el elemento en la posición `i` de esta lista contiene el número de veces que el establecimiento ha sido valorado con `i` estrellas.

Implementa en Python:

a) (2,5 PUNTOS) Un procedimiento, `actualizar_lista_establecimientos`, que tenga como parámetros el nombre de un fichero de texto como el descrito anteriormente y una lista de objetos de la clase `Establecimiento`. El procedimiento debe actualizar la lista dada (puede estar o no vacía), de forma que cada establecimiento distinto aparezca una única vez en la lista (supón que no habrá dos establecimientos distintos con el mismo nombre). El atributo `valoraciones` debe acumular la cantidad de veces que el establecimiento ha recibido cada una de las distintas valoraciones (de 0 a 5 estrellas). A partir de las líneas del fichero de ejemplo de arriba se obtendría la siguiente estructura:



***Nota**: Solo se permite recorrer el fichero una vez y recuerda que* `cadena.split('#')` *devuelve una lista con las subcadenas de* `cadena` *usando el carácter* `#` *como separador.*

b) (2,5 PUNTOS) Una función, `establecimientos_mejor_valorados`, que tenga como parámetro una lista de objetos como la usada en el apartado anterior y una localidad. La función debe construir y devolver una lista con los nombres de los establecimientos de la localidad dada que tengan más de 100 valoraciones en total y que hayan recibido más valoraciones de 5 estrellas que de todas las otras categorías juntas. Si ningún establecimiento de la localidad cumple estos requisitos, devolverá una lista vacía.

