**Exercicio 3 (5 punts)**

En este ejercicio codificaremos distintos métodos relacionados con el cálculo de distancias recorridas en diferentes trayectos entre ciudades y el consumo económico asociado al combustible usado en estos trayectos.

En la clase PAC1Ex3.java verás definida una matriz cuadrada de valores enteros (distanceMatrix). Esta matriz recoge las distancias (en kilómetros) entre una serie de ciudades origen y una serie de ciudades destino, según las rutas disponibles por carreteras. Así, cada fila corresponde a una ciudad origen, mientras que cada columna corresponde a una ciudad destino. Por tanto, cada valor representado por una fila y una columna de la matriz corresponde a la distancia en kilómetros entre la ciudad origen (fila) y la ciudad destino (columna). Por ejemplo, la posición[0][1](fila 0, columna 1) corresponde a la distancia entre la ciudad origen 0 (en este caso, Barcelona) y la ciudad destino 1 (en este caso, Hospitalet de Llobregat). Observa que:

* En código, no utilizamos los nombres de las ciudades, sino que, por simplicidad, simplemente nos referimos a él mediante el valor de fila y columna.
* Para valores iguales de fila y columna (p.e.,[0][0]) la distancia (valor) es 0, puesto que la distancia entre una ciudad y ella misma es siempre 0.
* La distancia entre una ciudad A y otra ciudad B no debe ser igual a la distancia entre la ciudad B y la ciudad A. Por ejemplo: la distancia en la posición[0][1]no es igual a la distancia en la ciudad posición[1][0]. Esto se debe a que las distancias medidas son trayectos a la carretera, y por tanto, algunas rutas en sentido contrario al original pueden ser más largas o más cortas que otras.

A continuación debes codificar los TODO. Este ejercicio consiste básicamente en codificar los 4 métodos/funciones siguientes:

* calculateTravelledDistance: este método calcula, para un listado de ciudades (representados por el número que representa cada ciudad en la matriz distanceMatrix), la distancia total recorrida desde la posición inicial de la ruta (es decir, la primera ciudad del listado) hasta la posición final de la ruta (es decir, la última ciudad del listado), pasando por todos los puntos intermedios. Por ejemplo, para la siguiente invocación:

calculateTravelledDistance({0,1,4,7})

El resultado de este método debería ser la suma de los siguientes trayectos:[0][1], [1][4]y [4][7]. Es decir, en este caso: 13+29+116=158.

Este método devuelve la distancia total recorrida a la ruta. Si la ruta recibida por parámetro está vacía, debe mostrar por pantalla el mensaje “[ERROR]: The route is empty” y devolver el valor 0. Si alguna de las ciudades recibidas por parámetro es un

valor no válido (es decir, un valor inferior a 0,o igual o superior al tamaño máximo de la matriz distanceMatrix), debe mostrar por pantalla el mensaje “[ERROR]: The route contains invalid destination points” y volver el valor -1.

* calculateTravelCostByMonth: esta función calcula el coste (en euros) asociado al consumo de combustible asociado a la distancia recorrida para una ruta específica. La función recibe 4 parámetros: (1) un listado de ciudades (route); (2) un valor entero (month) en el intervalo[0,11] que representa un mes del año (siendo 0 enero y

11 diciembre) en el que se ha realizado la ruta; (3) un valor decimal (literosPer100KM) que representa el consumo de combustible del vehículo con el que se ha realizado la ruta por cada 100 kilómetros; y (4) un valor booleano (isGasoline) que indica si el vehículo es de gasolina (true) o diesel (false). Dada esta información, el coste del trayecto es igual al total de litros de combustible consumidos por el vehículo (teniendo en cuenta su consumo por cada 100 kilómetros y la distancia total recorrida), multiplicados por el precio del litro de combustible en ese mes.



**IMPORTANTE:** Observa que te damos dos variables (arrays) llamadas GASOLINE y DIESEL. Cada una contiene el [precio medio de cada tipo de](https://www.dieselogasolina.com/Estadisticas/Historico) [combustible en Espanya](https://www.dieselogasolina.com/Estadisticas/Historico)  a día 1 del mes correspondiente a la posición del listado (para el 2022). Es decir: GASOLINE[0] contiene el precio por litro (en euros) de la gasolina a fecha 1 de enero de 2022; DIESEL[11] contiene el precio por litro (en euros) del diesel a fecha 1 de diciembre de 2022. Debes utilizar este valor para calcular el coste de combustible, según el mes indicado por parámetro.

Así, por ejemplo, la siguiente invocación:

calculateTravelCostByMonth({2,1,2}, 5, 3.9, false)

Calcula el coste asociado al recorrido entre los trayectos[2][1] y [1][2] realizados en el mes de junio (5) con un consumo de 3.9 litros por cada 100 kilómetros con un vehículo diesel (false). En ese caso, el coste total es de 3.69 euros. Ten en cuenta que esta función debe devolver -1:

* + si el valor del mes es inválido (es decir, si es inferior a 0 o superior a 11). En este caso, también debemos mostrar por pantalla el mensaje “[ERROR]: Invalid month”.
  + si el valor de litros por cada 100 kilómetros es inferior o igual a 0. En este caso, también debemos mostrar por pantalla “[ERROR]: Invalid litersPer100KM value”.
  + si alguna de les ciutats rebudes per paràmetre és un valor no vàlid (és a dir, un valor inferior a 0,o bé és igual o superior a la mida màxima de la matriu distanceMatrix)



**Consejo:** Utiliza la función calculateTravelledDistance que has codificado anteriormente.

* calculateTravelCosts: esta función calcula la suma total del coste asociado a un conjunto de trayectos (routes) realizados a lo largo de diferentes meses (months) con un mismo vehículo y, por tanto, para un mismo consumo (litersPer100KM) y un mismo tipo de combustible ( isGasoline). Cada trayecto a routes (es decir, cada fila de la matriz recibida como parámetro) corresponde a un trayecto realizado en el mes correspondiente a la misma posición del listado meses. Es decir, la ruta entre las ciudades representadas en routes[0] se ha realizado al mes identificado a months[0]. Si alguna de las rutas contiene un valor inválido, o bien si el mes o el valor del consumo a los 100 kilómetros son inválidos, el método debe devolver -1. De lo contrario, devuelve la suma total de costes.



**Consejo:** Utiliza la función calculateTravelCostByMonth que has codificado anteriormente.

* isGasolineCheaper: esta función calcula, para un conjunto de trayectos (routes) realizados a lo largo de diferentes meses (months), el coste asociado al consumo realizado por un coche de gasolina (litersPer100KMGasoline) respecto a los mismos trayectos realizados por un coche diesel (litersPer100KMDiesel). Si la opción del coche de gasolina hubiera resultado más económica, el método devuelve true. De lo contrario, devuelve false.



**Consejo:** Utiliza la función calculateTravelCosts que has codificado anteriormente.