INTRODUCCIÓN

Este documento es el conjunto de condiciones y tareas para el proyecto por grupos para la Evaluación Continua de la asignatura Fundamentos de Informática en la Escuela de Ingeniería de Vitoria, UPV/EHU.

REGLAS GENERALES

- Los grupos deben seer como máximo de **tres** personas, y deben trabajar cada uno sin ayuda de otro grupo. Los grupos que presenten código sospechosamente parecido suspenderán el proyecto y la asignatura en su totalidad.
- Se debe subir a eGela la última versión que se tenga del trabajo **por lo menos una vez por semana**. Tened en cuenta de que hacer grandes cambios en poco tiempo es sospechoso para el profesorado. Para evitar problemas, subid el código a menudo para que se pueda ver la evolución de vuestro trabajo (ver sección Subir el Proyecto).

EVALUACIÓN POR GRUPOS

Cada grupo tendrá una calificación que será la base de la nota de cada alumno (que puede ser mayor o menor que la del grupo). La nota máxima del grupo dependerá del número de tareas finalizadas correctamente por el grupo:

- 100 %: todas las tareas
- 75 %: tareas 1, 2, 3, 4 y 6.

DIRECTRICES PARA LA PROGRAMACIÓN

- No imprimir mensajes por pantalla, crear gráficas o pedir entradas dentro de una **función**, a no ser que sea el objetivo de dicha función. La entrada y salida de datos de las funciones es mediante parámetros y resultado.
- Cuando sea posible, intenta reutilizar código mediante llamadas a funciones. En la mayor parte de las tareas se indicará cómo diseñar las funciones.
- Elige nombres con significativos y con la nomenclatura adecuada para crear variables/*scripts*/funciones, de modo que cualquiera que lea el código pueda entenderlo y seguir el valor de las variables sin pasar tiempo averiguando qué es cada elemento.
- Usa ciclos (`for/while`) para iterar (repetir instrucciones) sobre un conjunto de valores en lugar de repetir código ya existente.
- Escribe comentarios (con el símbolo de porcentaje %) para explicar el objetivo de cada parte de los programas y qué datos almacenan las variables.

SUBIR EL PROYECTO

El proyecto se entrega con las siguientes normas:

1. Las entregas se realizarán a través de una tarea de eGela con el formato y frecuencia que se detallan más adelante. No se admitirán otros medios salvo causa de fuerza mayor.
2. Con la intención de facilitar la revisión de vuestro trabajo, observar la evolución y evitar el fraude, os solicitamos que subáis el código a la tarea de *eGela* habilitada al respecto, **al menos una vez a la semana**.
3. Para comprobar la evolución del proyecto **no borréis** versiones antiguas.
4. Tan sólo una persona integrante del grupo debe subir el código.

5. Cada vez que se suba el código es obligatorio subir un fichero `ZIP`:

   a) El nombre del fichero debe tener exactamente el formato siguiente: `'project-`*yyyy*`-`*mm*`-`*dd*`.zip'`, donde *yyyy* es el año, *mm* el mes y *dd* el día.

   b) El fichero `zip` debe contener todos los *scripts* y funciones que se hayan desarrollado hasta la fecha.

   c) El fichero `zip` **no debe** contener **ni ficheros de datos ni los resultados** solicitados en las tareas que hayáis podido desarrollar hasta ese punto (gráficas, ficheros de texto, salidas por pantalla).

   d) Dentro del fichero comprimido **también** se incluirá un fichero **único** llamado `'log.txt'`, donde se informará de lo realizado durante de desarrollo del proyecto. Este fichero irá creciendo con las aportaciones del trabajo realizado cada semana.

   No se deben borrar las partes ya entregadas en un `zip` anterior, sino que el último `zip` entregado debe contener la totalidad de ficheros generados en las tareas del proyecto.

   Por ejemplo:

   log.txt en `project-2022-12-03.zip`

   2022-11-29
   Tarea 1: Comenzamos a programar el script 'myscript.m', pero todavía no funciona. Aitor hace la parte de carga de ficheros, Blanca la selección de columnas y Jon la del plot.

   2022-12-03
   Tarea 1: Ya lo hemos terminado de revisar, había un error en el bucle principal que ha encontrado Blanca.

   Task 2: Hemos comenzado a escribir en clase la función 'myfunction'

OBJETIVO DEL PROYECTO

La información meteorológica tiene una gran influencia en nuestras vidas: cosechas, energías renovables, agua, peligros… Los datos se obtienen de radares, satélites y estaciones terrestres.

En este proyecto vamos a procesar los datos recogidos por cuatro estaciones meteorológicas terrestres en un año. La información es un conjunto medidas de temperaturas,precipitaciones y velocidad del viento.

En esta práctica se debe completar el proyecto implementando las funcionalidades indicadas en el apartado Tareas.

FICHEROS DE ENTRADA

En este proyecto se procesarán los ficheros incluidos en el fichero denominado `ProjectFiles.zip` y por ello lo descomprimiremos en la carpeta del proyecto, dentro de una carpeta llamada `data`. De esta forma tendremos en dicha carpeta 192 ficheros que contienen los datos meteorológicos recogidos en cinco estaciones meteorológicas. Los nombres de fichero están estandarizados y siguen el patrón:

identificador`_`*medida*`_`*añomes*`.csv`

donde:

- *identificador*: código alfanumérico asociado a la estación meteorológica.
- *medida*: una cadena que representa la medida muestreada.
- *año*: número de cuatro cifras que corresponde al año en el que se obtuvieron los datos del fichero.
- *mes*: número de cuatro cifras que corresponde al mes en el que se obtuvieron los datos del fichero.

Tenemos información recogida en las siguientes estaciones meteorológicas:

Código	Estación meteorológica
C042	Punta Galea
C057	Mungia
C061	Arboleda
C067	Gardea

Tabla 1: Estaciones meteorológicas y sus códigos identificativos

En este proyecto tendremos en cuenta tres medidas diferentes:

- **Temperatura del aire** (*AirTemp*), en grados centígrados o Celsius ($^{\circ}C$). Los ficheros que contienen la temperatura del aire la organizan con la siguiente información:
  - *Year*: año en el que se midió la temperatura del aire.
  - *Month*: mes en el que se midió la temperatura del aire.
  - *Day*: día en el que se midió la temperatura del aire.
  - *Hour*: hora en el que se midió la temperatura del aire.
  - *Minute*: minuto en el que se midió la temperatura del aire.
  - *Second*: segundo en el que se midió la temperatura del aire.
  - *AirTemo*: temperatura del aire.
  - *DataOrigin*: identificador del sensor que midió la información.

Ejemplo: *C042_AirTemp_202101.csv*

```
Year,Month,Day,Hour,Minute,Second,AirTemp,DataOrigin
2021,1,1,0,0,0,6.95,0
2021,1,1,0,10,0,6.89,0
2021,1,1,0,20,0,6.85,0
...
```

- **Velocidad del viento** (*WindSpeed*), en metros por segundo (m/s). Los ficheros que contienen la velocidad del viento la organizan con la siguiente información:

- *Year*: año en el que se midió la velocidad del viento.
- *Month*: mes en el que se midió la velocidad del viento.
- *Day*: día en el que se midió la velocidad del viento.
- *Hour*: hora en el que se midió la velocidad del viento.
- *Minute*: minuto en el que se midió la velocidad del viento.
- *Second*: segundo en el que se midió la velocidad del viento.
- *WindSpeed*: velocidad del viento en ese instante.
- *WindDirection*: dirección del viento en ese momento. La dirección se representa en grados, donde 0 grados representa el *Norte*, 90 grados representa el *Este*, 180 grados representa el *Sur* and 270 grados representa el *Oeste*.
- *DataOrigin*: identificador del sensor que midió la información.

Ejemplo: *C042_WindSpeed_202101.csv*

```
Year,Month,Day,Hour,Minute,Second,WindSpeed,WindDirection,DataOrigin
2021,1,1,0,0,0,7.178,0,0
2021,1,1,0,10,0,7.09,0,0
2021,1,1,0,20,0,6.233,0,0
...
```

- **Precipitaciones** (*Precipitation*), medida en litros por metro cuadrado (l/m^2) (también llamada milímetros porque un litro es un metro por un metro por un mm). Los ficheros que contienen las precipitaciones la organizan con la siguiente información:

  - *Year*: año en el que se midió la precipitación.
  - *Month*: mes en el que se midió la precipitación.
  - *Day*: día en el que se midió la precipitación.
  - *Hour*: hora en el que se midió la precipitación.
  - *Minute*: minuto en el que se midió la precipitación.

- *Second*: segundo en el que se midió la precipitación.
- *WindSpeed*: precipitación en ese instante.
- *Precipitation*: precipitación desde el instante previo.
- *DataOrigin*: identificador del sensor que midió la información.

Ejemplo: *C042_Precipitation_202101.csv*

```
Year,Month,Day,Hour,Minute,Second,Precipitation,DataOrigin
2021,1,1,0,0,0,0.1,0
2021,1,1,0,10,0,0.0,0
2021,1,1,0,20,0,0.0,0
...
```

- **Humedad relativa** (*RHumidity*), que representa la proporción de la cantidad de vapor de agua presente en el aire, medida en un porcentaje ([0, 100]). Los ficheros que contienen la humedad relativa presentan esta estructura:

  - *Year*: año en el que se midió la humedad relativa.
  - *Month*: mes en el que se midió la humedad relativa.
  - *Day*: día en el que se midió la humedad relativa.
  - *Hour*: hora en el que se midió la humedad relativa.
  - *Minute*: minuto en el que se midió la humedad relativa.
  - *Second*: segundo en el que se midió la humedad relativa.
  - *RHumidity*: humedad relativa.
  - *DataOrigin*: identificador del sensor que midió la información.

Ejemplo: *C042_RHumidity_202101.csv*

```
Year,Month,Day,Hour,Minute,Second,RHumidity,DataOrigin
2021,1,1,0,0,0,81.6,0
2021,1,1,0,10,0,82.0,0
2021,1,1,0,20,0,82.7,0
...
```

Nota: Aunque las medidas fueron grabadas en ficheros separados, se guardaron de forma síncrona, es decir, los instantes de tiempo (año, mes, día, hora, minuto y segundo) son los mismos en todos los ficheros que contienen las diferentes medidas grabadas en una estación en un cierto mes.

TAREAS

Tarea 1: Visualizar los datos meteorológicos durante el año

En esta primera tarea vamos a analizar los datos meteorológicos recogidos por las cinco estaciones mencionadas durante el año 2021. Para ello, debéis implementar un programa en Matlab/Octave. El *script* llamado `examineWeatherData` analizará la temperatura del aire, precipitaciones y la velocidad del viento. La información de los ficheros se mostrará en una figura que contenga tres gráficas (ver Figura 1). La primera gráfica debe mostrar la evolución de las temperaturas medias mensuales durante el año. La segunda gráfica mostrará las precipitaciones medias diarias, y en la última se graficará la velocidad media del viento.

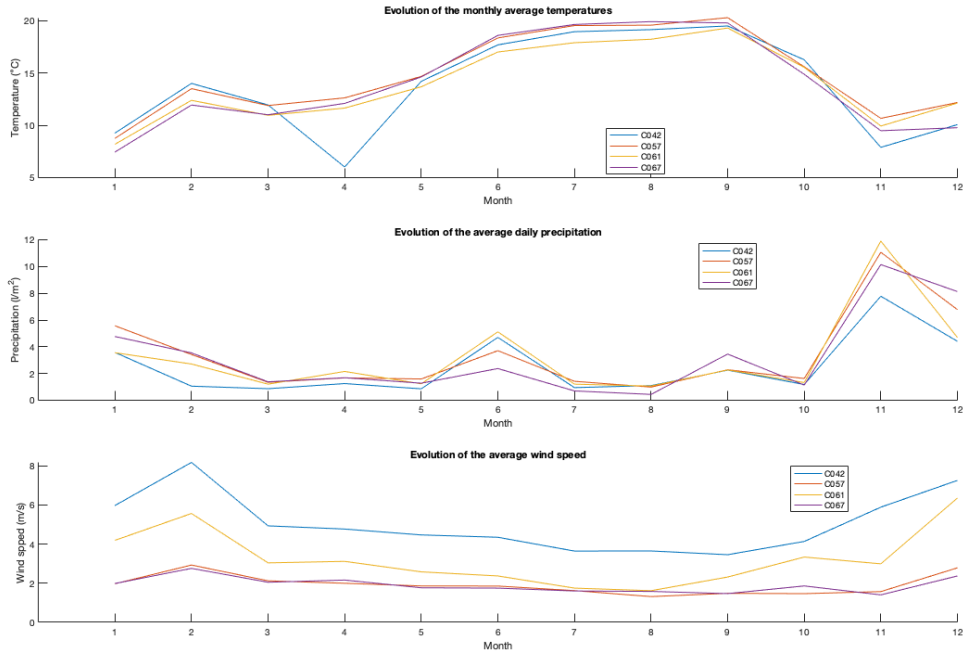

Figura 1: Figura que muestra la evolución de los datos meteorológicos durante el año

Nota: La información sobre precipitaciones se almacena en cada momento como la precipitación acumulada desde la medición anterior. La precipitación media diaria se puede calcular sumando las precipitaciones acumuladas durante el mes y dividiendo entre el número de días.

Tarea 2: Análisis de las precipitaciones

Las precipitaciones son necesarias para agricultura, energía hidráulica y para los ecosistemas, pero pueden ser un factor de riesgo por riadas e inundaciones o avalanchas. Debéis desarrollar un *script* llamado `analyzePrecipitations` en esta tarea, que calcule y muestre la siguiente información por pantalla:

1) Precipitaciones medias
2) Desviación típica de las precipitaciones
3) Número de días sin precipitaciones
4) Días con precipitaciones extremas

Para esto, debéis completar los siguientes pasos:

1. Implementad una función llamada `computeDailyPrecipitations` que, dado el vector que contiene las precipitaciones recogidas en una estación meteorológica durante un mes, devuelva un vector con las precipitaciones diarias. En este vector que devuelve la función cada día del mes contiene la precipitación caída en el día correspondiente.

   ```
   function dPrep = computeDailyPrecipitations(pData)
   ```

   Puedes comprobar que la función es correcta ejecutando el siguiente código y comprobando que se producen los mismos resultados.

   ```
   precipD = [2021 1 1 0  0  0 12 0
              2021 1 1 0 10 10  1 0
              2021 1 2 0  0  0  1 0];
   dPrec = computeDailyPrecipitations(precipD)

   dPrec = 13     1
   ```

2. Implementad una función llamada `computePrecipitationStats` que, dado el vector que contiene las precipitaciones recogidas en una estación meteorológica durante un mes y un valor escalar que define un umbral (*threshold*),

devuelva la precipitación diaria media, la desviación típica de las precipitaciones diarias, el número de días sin precipitaciones y un vector que contenga los días en los que la precipitación superó el umbral especificado.

```
function [avgP,stdP, dWP, ePD] =
    computePrecipitationStats(pData, thresh)
```

Puedes comprobar que la función es correcta ejecutando el siguiente código y comprobando que se producen los mismos resultados.

```
[av, st, wp, ep] = computePrecipitationStats([13 1 0],10)

av = 4.6667
st = 7.2342
wp = 1
ep = 1
```

3. Implementad en Matlab/Octave el programa o *script* `analyzePrecipitations` que genera un informe que muestre los valores estadísticos de las estaciones meteorológicas con el siguiente formato:

```
Month: 01
=========
Station C042:
Avg. Precipitation: 3.58 l/m^2
Precipitation Std.: 5.10 l/m^2
Number of days without precipitation: 12
Days with extreme precipitations:         None
...
Station C067:
Avg. Precipitation: 8.13 l/m^2
Precipitation Std.: 12.55 l/m^2
Number of days without precipitation: 17
Days with extreme precipitations:          9
```

Además, el *script* debe generar para cada mes una figura que compare las precipitaciones diarias para cada estación meteorológica mediante gráficos

de barras (ver Figura 2). Se añadirá una línea horizontal que represente la precipitación media. Los días en los que la precipitación se considere extrema (más de $40l/m^2$) se mostrarán en rojo.

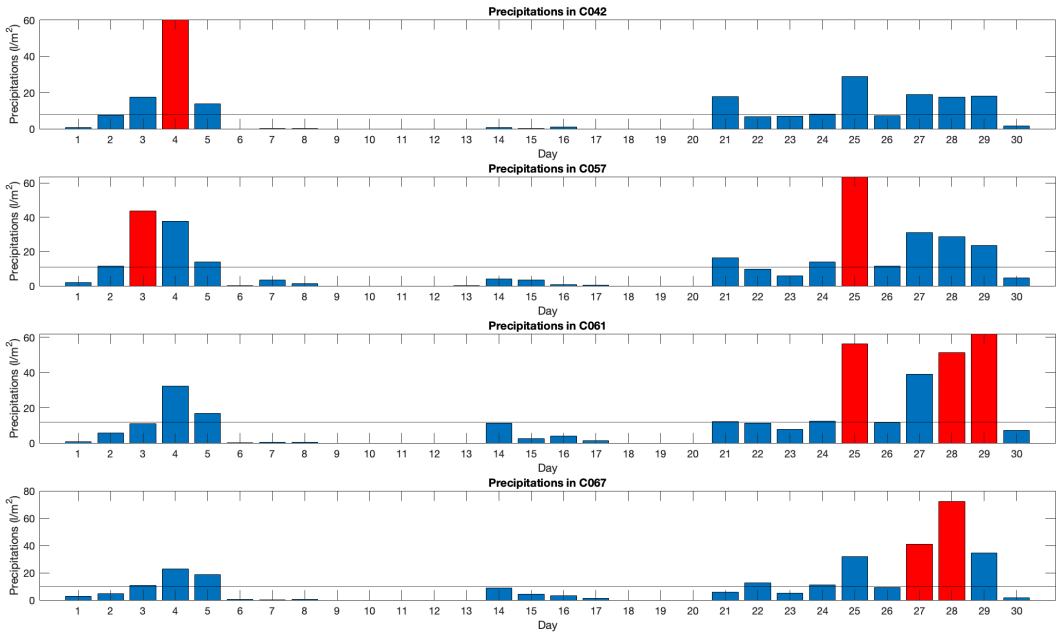

Figura 2: Figura que compara las precipitaciones de un mes en cada estación

Ayuda: el color de barras concretas se puede cambiar de esta forma:

```
%Crear y guardar el gráfico en la variable b
b = bar(values);

%cambiar color de la segunda barra del gráfico b
b.FaceColor = 'flat';
b.CData(2,:) = [1 0 0]; %rojo en RGB
```

Se pueden añadir líneas horizontales a un gráfico con el siguiente código:

```
yline(value, format);
```

donde *value* sería la precipitación media y *format* el tipo de línea (por ejemplo, `'-k'` para una línea negra).

Tarea 3: Análisis de distribución de temperaturas

Esta zona de clima oceánico se caracteriza por temperaturas moderadas, humedad constante y chubascos frecuentes. De todas formas, en los últimos años se muestra el cambio climático en el incremento de temperaturas. Por ello se va a realizar en esta tarea un análisis de temperaturas durante el año. En particular, se estudiará la distribución en diferentes rangos de temperaturas. Los rangos se muestran en la tabla Tabla 2.

Zona	Descripción	Rango de temperatura
1	Bajo cero	<0
2	Frío	[0, 10)
3	Moderado	[10, 15)
4	Templado	[15, 20)
5	Cálido	[20, 25)
6	Extremadamente cálido	>25

Tabla 2: Rangos de temperaturas predefinidos

Para ello, debéis completar los siguientes pasos:

1. Implementad una función denominada `toSeconds` que, dados los vectores que representan horas, minutos y segundos de los tiempos de las mediciones, devuelva un vector que contenga los tiempos en segundos.

   Si los vectores de entrada tienen diferente número de elementos, la función devuelve un vector vacío.

   ```
   function timestamps = toSeconds(hours, mins, secs)
   ```

   Se puede comprobar la corrección de la función ejecutando el siguiente código y comprobando los resultados obtenidos.

   ```
   hours = [1 2 10];
   minutes = [0 10 2];
   seconds = [5 10 13];

   timestamps = toSeconds(hour, minutes, seconds)
   timestamps = 3605        7810        36133
   ```

2. Implementad una función denominada `analyzeTemperatureRanges` que, dado un vector que representa los segundos de las mediciones y un vector que contiene las temperaturas, devuelva un vector que contenga el tiempo que la temperatura ha pasado en cada uno de los rangos definidos en la tabla Tabla 2.

   ```
   function tempZones = analyzeTemperatureRanges(
       timeStamps, temps)
   ```

   Se puede comprobar la corrección de la función ejecutando el siguiente código y comprobando los resultados obtenidos.

   ```
   timestamps = [0 1200 3600];
   temps = [10 -5 32];

   tempZones = analyzeTemperatureRanges(timestamps, temps)
   tempZones = 724  1448  326  324  324  454
   ```

   Pista: La temperatura es una medida que cambia de forma continua. Para obtener unos resultados más precisos, se debe aplicar interpolación con un $\Delta t = 1$s.

3. Implementad un programa principal o *script* llamado *analyzeTemperatures* que procese los ficheros de temperaturas y que, por cada mes, genere un fichero `png` que contenga una gráfica resumen de la distribución de temperaturas (en los rangos de temperatura especificados anteriormente) por cada estación meteorológica (ver Figura 3). Los ficheros se deben guardar en la carpeta *Reports*. La distribución se mide en porcentaje de tiempo.

   Los nombres de los ficheros generados deben seguir el formato:

   `Temperature_`*`Year`*`_`*`Month`*`.png`

   donde *Year* es un número de 4 cifras que representa el año y *Month* es un numero de 2 cifras del mes correspondiente a la información mostrada en la imagen.

Figura 3: Distribución de temperaturas en los rangos de temperaturas

Indicación:

Como se puede observar en los ficheros de datos, la última medida se realiza a las 23:50:00, por lo que los últimos diez minutos del día no son tenidos en cuenta. Para evitar este problema, la medida se puede estimar o *extrapolar* usando la función `interp1` usando este código:

```
%Add a new entry for temperatures and timestamps
lastTimeStamp = 24 * 3600;
%'linear' and 'extrap' must be used to infer the
%value for a timestamp out of the range
lastTemp = interp1(timestamps,temperatures,
   lastTimeStamp,'linear','extrap');
timestamps(end + 1) = lastTimeStamp;
temperatures(end + 1) = lastTemp;
```

Los gráficos de barras *apilados* se generan con la opción `'stacked'`:

```
bar(values, 'stacked');
```

Para evitar que la leyenda se ubique sobre las barras del gráfico, la colocamos fuera con la opción `'eastoutside'`:

```
legend(legendNames, 'location', 'eastoutside');
```

Tarea 4: Análisis de la sensación térmica

La *sensación térmica* es un término usado para describir cómo se percibe la temperatura en la piel humana debido a la combinación de la propia temperatura con la humedad y el viento a los que estamos expuestos. El viento afecta a la sensación térmica cuando las temperaturas son bajas (entre $-50\,°C$ y $10\,°C$) y la velocidad del viento está entre $4.8\,km/h$ y $177\,km/h$ o cuando las temperaturas son altas (sobre $26\,°C$) y la humedad sobre $40\,\%$.

La sensación térmica se puede calcular así:

- Cuando la temperatura está en el rango [-50, 10] y **simultáneamente** la velocidad de viento está en el rango [4.8, 177] se aplica la siguiente fórmula:

$$WC = 13{,}1267 + 0{,}6215 * T - 11{,}37 * S^{0{,}16} + 0{,}3965 * T * S^{0{,}16}$$

  donde:

  - WC es la sensación térmica en grados Celsius.
  - T es la temperatura medida en grados Celsius.
  - S es la velocidad del viento en **km/h**.

- Cuando la temperatura es alta (más de $26\,°C$) y la humedad es mayor de $40\,\%$ aplicaremos la fórmula:

$$\begin{aligned} WC = & -8{,}78469476 + 1{,}61139411 * T + 2{,}338548839 * H \\ & - 0{,}14611605 * T * H - 0{,}012308094 * T^2 - 0{,}016424828 * H^2 \\ & + 0{,}002211732 * T^2 * H + 0{,}00072546 * T * H^2 \\ & - 0{,}000003582 * T^2 * H^2 \end{aligned}$$

donde:

- WC es la sensación térmica en grados Celsius.
- T es la temperatura medida en grados Celsius.
- H es la humedad relativa del aire en porcentaje.

- En cualquier otro caso la sensación térmica tiene el mismo valor que la temperatura medida en la estación meteorológica.

En esta tarea debéis implementar un programa en Matlab/Octave que analice el efecto del viento y la humedad en la sensación térmica, completando los pasos:

1. Implementad una función denominada `computeWindChill` que, dado tres vectores que contienen las temperaturas, velocidad del viento y humedad, devuelva un vector que contenga las temperaturas que representan la sensación térmica.

   ```
   function wc = computeWindChill(temp, wpeed, rHum)
   ```

   Se puede comprobar la corrección de la función ejecutando el siguiente código y comprobando los resultados obtenidos.

   ```
   times = [0 1200 3600];
   temperature = [-5 15 29];
   windspeed = [25 0 89];
   humidity = [70 65 90];

   wc = computeWindChill(temperature, windspeed, humidity)

   wc = -12.3285   15.0000   37.0756
   ```

2. Implementad una función denominada `computeWindChillMAD` que, dados tres vectores que contienen los tiempos, las temperaturas en ese segundo y las temperaturas que representan la sensación térmica, devuelva la *diferencia media absoluta* (MAD) entre temperaturas del aire y las temperaturas de sensación térmica.

```
function mad = computeWindChillMAD(timestamps,
    temps, wcTemps)
```

La MAD se calcula mediante la fórmula:

$$mad = \frac{\sum_{i=1}^{T-1} |t_i - wc_i| * \Delta t_i}{\sum_{i=1}^{T-1} \Delta t_i}$$

donde t_i es la temperaturas del aire en el momento de medición i, wc_i es la temperatura de sensación térmica en el momento i, Δt_i es la duración del periodo desde el momento i al momento $i+1$, y T es el número de momentos en los que se realizan mediciones. **Observad que t_i se considera constante desde una medición hasta la siguiente.**

Se puede comprobar la corrección de la función ejecutando el siguiente código y comprobando los resultados obtenidos.

```
times = [0 1200 3600];
temperature = [-5 15 29];
wcTemperature = [-12.3285 15.0000 37.0756];

mad = computeWindChillMAD(times, temperature, wcTemperature)

mad = 2.4428
```

3. Desarrollad un *script* llamado `analyzeWindChill` que analiza los datos meteorológicos y crea una gráfica que muestra la evolución de la MAD mensual (ver Figura 4). La MAD mensual es la media de las MAD diarias.

Además el *script* debe imprimir un informe que indique para cada estación meteorológica cuál es la MAD mensual máxima y el mes correspondiente. El formato será el siguiente:

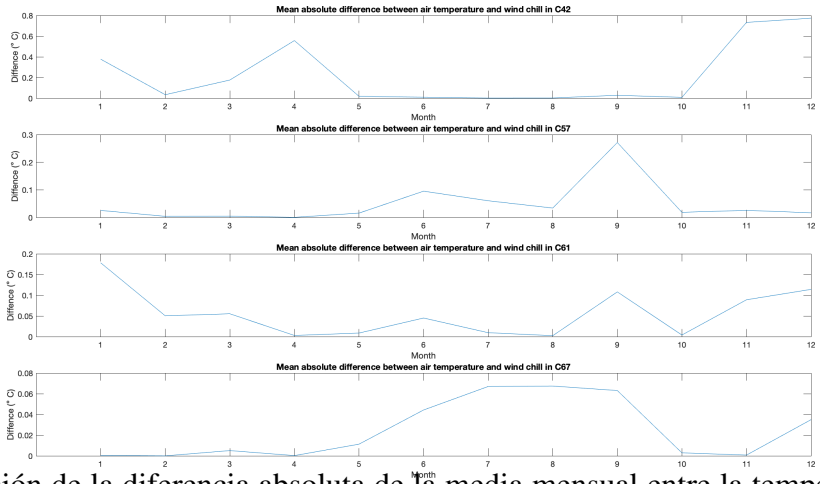

Figura 4: Evolución de la diferencia absoluta de la media mensual entre la temperatura del aire y la sensación térmica

```
Mean absolute difference between air temperature and wind chill
report

Station C042
Max. MAD: 0.77 Month: December

Station C057
Max. MAD: 0.27 Month: September

Station C061
Max. MAD: 0.18 Month: January

Station C067
Max. MAD: 0.07 Month: August
```

Tarea 5: Estimación de la capacidad de generación de energía

Un aerogenerador es un dispositivo que convierte la energía cinética del viento en energía eléctrica. Supongamos que se puede estimar la energía eléctrica que puede generar mediante la siguiente fórmula:

$$E_w = A * 0{,}5 * \rho * v^3 * t / 10^9$$

donde:

- E_w es la energía del viento en gigajulios (GJ).
- A es el área de flujo del aire m^2 que se puede calcular con la fórmula:

  $$A = \pi * r^2$$

  siendo r la longitud de las palas del aerogenerador en metros (ver Figura 5).
- ρ tiene el valor $1{,}225 kg/m^3$ y es la densidad del aire[1].
- v es la velocidad del viento en m/s.
- t es el tiempo en segundos.

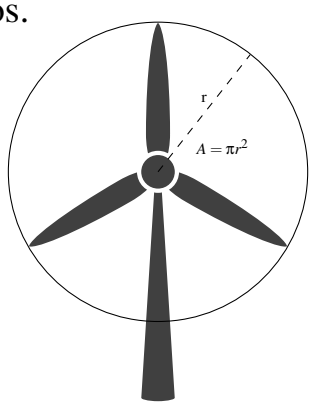

Figura 5: Longitud de las palas del aerogenerador (r) y área del flujo de aire (A)

Este prototipo de aerogenerador no gira hacia diferentes direcciones para encarar el viento, y por lo tanto la energía eléctrica que puede producir depende de la

[1] En realidad, $\rho = 1.225\,\mathrm{kg/m^3}$ a la presión de 1013.25 hPa y 15° de temperatura, pero para esta tarea vamos a asumir que es el valor correcto y constante para todas las estaciones.

orientación en la que se instala y de la dirección del viento. Ésta se suele indicar con un ángulo, más manejable que los puntos cardenales.

Podemos resumir la correspondencia de los puntos cardinales[2] con los rangos de grados en la Figura 6 y Tabla 3.

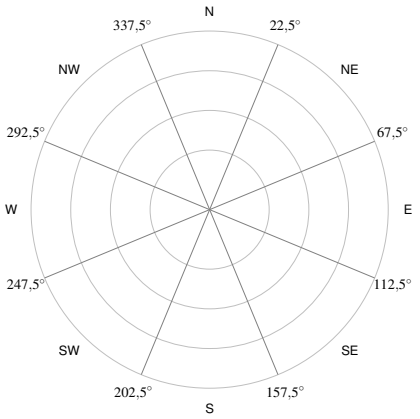

Figura 6: Direcciones del viento

Dirección del viento	Rango
Norte	[337, 22.5]
Noreste	(22.5, 67.5)
Este	[67.5, 112.5]
Sureste	(112.5, 157.5)
Sur	[157.5, 202.5]
Suroeste	(202.5, 247.5)
Oeste	[247.5, 292.5]
Noroeste	(292.5, 337.5)

Tabla 3: Direcciones del viento

Queremos analizar la idoneidad de las áreas donde están ubicadas las estaciones meteorológicas para la generación de energía eólica. Vamos a considerar que podemos situar ocho aerogeneradores alrededor de la estación, uno orientado de forma fija en cada dirección (*Norte*, *Noreste*, *Este*, *Sureste*, *Sur*, *Suroeste*, *Oeste* y *Noroeste*). En un primer análisis elegimos un tamaño de pala de 20 metros. El objetivo principal de esta tarea es estimar cuánta energía eólica se puede generar en un año en cada dirección alrededor de cada estación.

Vamos a llevar a cabo los siguientes pasos:

1. Implementad una función denominada `computeWindEnergy` que, dado un vector que contiene los tiempos en segundos en los que se han medido la velocidad del viento, un vector con la velocidad del viento, un vector con la dirección del viento (en grados) para cada velocidad, un escalar con la longitud de las aspas, devuelva el total de energía eólica (en gigajulios) que se generaría en cada uno de los 8 aerogeneradores, en función de su orientación.

   Si los vectores de entrada tienen diferente longitud, el vector de ocho elementos de salida de la función estará compuesto por ceros.

[2] *Norte*, *Noreste*, *Este*, *Sureste*, *Sur*, *Suroeste*, *Oeste* y *Noroeste*.

```
function windEnDir = computeWindEnergy(timeslices,
    windspeed, winddirection, r)
```

Se puede comprobar la corrección de la función ejecutando el siguiente código y comprobando los resultados obtenidos.

```
times=10:20:50;
wSpeed=[200 200 250];
wDirection=[340 67.5 247.4];
R=20;

windEnDir=computeWindEnergy(times,wSpeed,wDirection, R)

windEnDir = 123.1504   0   123.1504   0   0   240.5282   0   0
```

2. Implementad un *script* llamado `analyzeWindEnergy` que cree un informe para cada estación meteorológica con el valor máximo anual de energía eólica que se podría generar junto con la dirección en la que debería estar orientado el aerogenerador, en el siguiente formato:

```
Station: C42
- Maximun Wind Energy: 3053.66 (GJ)
- Wind turbine faced North west
Station: C57
- Maximun Wind Energy: 255.28 (GJ)
- Wind turbine faced North west
Station: C61
- Maximun Wind Energy: 2314.15 (GJ)
- Wind turbine faced South west
Station: C67
- Maximun Wind Energy: 320.48 (GJ)
- Wind turbine faced South west
```

El programa también debe crear una gráfica que nos permita comparar la energía anual generada por cada dirección y estación meteorológica en un gráfico de barras (ver Figura 7).

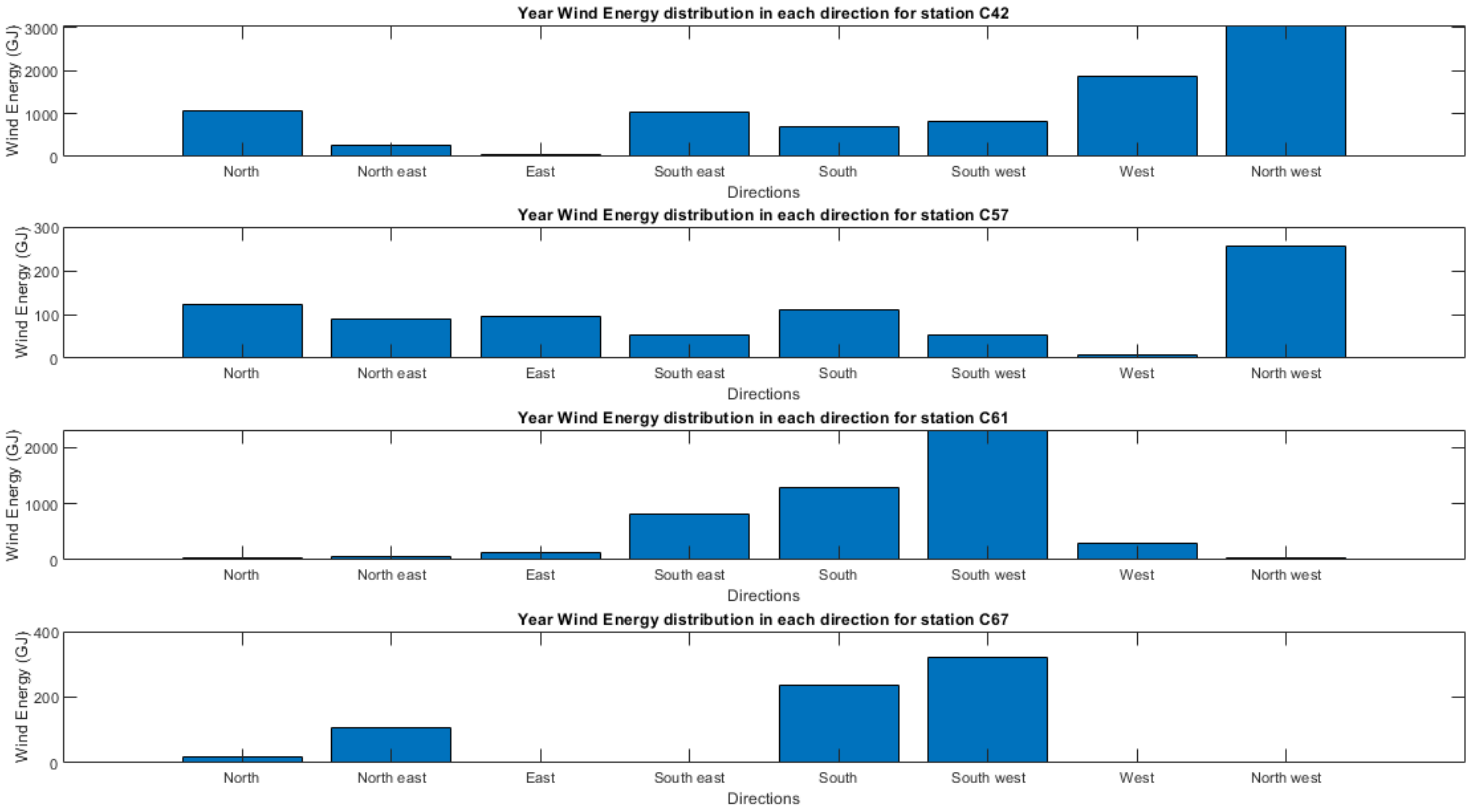

Figura 7: Gráfica que compara la energía anual generada por dirección en las cuatro estaciones.

Tarea 6: Menú del programa principal

Implementad el *script* `mainMenu.m` que muestre el siguiente menú.

```
###### MENU ######
1. Visualizar la evolución de datos meteorológicos durante el año
2. Análisis de las precipitaciones
3. Análisis de distribución de temperaturas
4. Análisis de la sensación térmica
5. Estimación de la capacidad de generación de energía
0. Salir
###### MENU ######
Selecciona una opción:
```

Cada entrada del menú se corresponde con una de las tareas realizadas en este proyecto. Según la opción elegida, el programa ejecutará el *script* de la tarea correspondiente. Para ejecutar un *script* desde otro *script* sólo hace falta escribir su nombre sin la extensión `.m`. Si el número introducido como entrada no es correcto, el *script* mostrará un mensaje de error: `La opción elegida no es correcta: elige una opción entre 1 y 5, o 0 para salir`.

Antes de mostrar el menú, el *script* limpiará la consola y el espacio de variables (`clc, clear`). Al elegir una de las opciones, se mostrarán los resultados de la tarea correspondiente. Después de mostrar los resultados de una tarea y antes de mostrar el menú de nuevo, se indicará a la persona usuaria que debe pulsar alguna tecla mediante un mensaje: "`Pulsa una tecla para continuar...`" para poder ver los resultados con la instrucción `pause`. A continuación se mostrará de nuevo el menú.

Esta operación se repetirá mientras no se elija la opción *0. Salir*. No uses la función `exit()` ya que cierra Matlab/Octave. Sólo es necesario que termine la estructura iterativa que repite el menú hasta que se le introduzca cero.