La siguiente práctica se debe realizar **individualmente**, y su autoría no debe estar en ningún caso en entredicho con la de otros estudiantes o con materiales de la red. Todas las prácticas pasarán por sistemas antiplagio. **Dada la facilidad de copia de este tipo de tarea, se asumirá su copia bajo la mínima sospecha, conllevando un 0 a todos los implicados.**

El propósito de esta actividad es que vosotros desarrolléis vuestras propias habilidades de creación de código, una habilidad que no se ve desarrollada si no sois vosotros los que implementáis la solución.

Para la realización de los ejercicios **no se deben usar funciones o librerías no vistas en clase**, en especial no se puede usar `min`, `max`, `sum`, `count`, `sorted`.

Evaluación

Cada ejercicio tiene una nota asignada en puntos siendo el máximo un 10.

Cada ejercicio tiene un set de tests públicos (asserts) para comprobar que su funcionamiento es correcto, estos asserts os ayudarán durante el desarrollo de la práctica para comprobar que lo que estáis implementando cumple con lo que se pide. Pero, además, **al corregir la práctica vuestras funciones serán testeadas con otro conjunto de asserts privados**, por ello debéis leer con atención el enunciado ya que los asserts públicos no comprueban exhaustivamente todos los casos. **Todo código entregado sin su archivo replay correspondiente no será corregido.**

Entrega

Para realizar la entrega la debéis realizar en https://www.codereplay.dev/, al finalizar cada ejercicio debéis pulsar dos botones "save file" y "save replay". Save file, os guardará vuestro código final, y save replay guardará como habéis generado este código. Se deben entregar ambos ficheros para cada ejercicio, renombrando los ficheros con el nombre del ejercicio, por ejemplo saludos_email.py y saludos_email.json. **Todo código entregado sin su archivo replay correspondiente no será corregido.**

(3.5p)

Dada una matriz m (no necesariamente cuadrada) de usuarios donde cada fila tiene un usuario con su identificador, nombre, edad, y costo de las compras en la plataforma, crea las siguientes funciones:

media_edad(m). Retorna la media de edad en la plataforma media_de_compras_usuario(u). Dado un usuario retorna la media del costo de sus compras. max_de_medias(m). Retorna el usuario la media de costo de compras más alta. rango_de_edad(m). Retorna el mínimo y el máximo de edad de los usuarios de la plataforma.

Como siempre no se pueden usar funciones min, max, etc. No implementadas por vosotros.

In [ ]:
```python
def media_edad(m):
    pass

def media_de_un_usuario(usuario):
    pass


def max_de_medias(m):
    pass


def rango_de_edad(m):
    pass

usuarios = [['1234567D','Joan',50, 210, 20, 40],
            ['4324D','Antonio',69,710,550,400,698,645,512],
            ['5544C','Claudio',72,100,534],
            ['5432F','Maria',75,770,645,230,650,590,481,602]]

assert(media_edad(usuarios) == 66.5)
assert(media_de_un_usuario(['1234567D','Joan',50, 210, 20, 40]) == 90)
assert(max_de_medias(usuarios) == "Antonio")
assert(rango_de_edad(usuarios) == (50, 75))
```

(3.5p)

Dada una matriz con datos de alumnos en una asignatura: id, nota1, nota2, nota3 y nota4. (Todas las notas tienen el mismo peso). Crea las siguientes funciones:

aprobados(m): retorna una lista con los identificadores de los alumnos que han aprovado.

alumno_mejor_nota(m): retorna el id del alumno con mejor en cualquiera de las pruebas.

alumno_mejor_nota(m): retorna el id del alumnos con mejor nota media.

alumnos_peores_notas(m,n): retorna una lista con los identificadores de los *n* alumnos con notas medias más bajas.

In [ ]:
```python
def aprobados(m):
    pass

def alumno_mejor_nota(m):
    pass


def alumno_mejor_nota_media(m):
    pass


def alumnos_peores_notas(m,n):
    pass




list_notas = [  ['33333R', 0.0, 9.0, 8.5, 6.0],
                ['337464F', 8.5, 7.0, 2.0, 8.5],
                ['00000F', 0.5, 1.0, 2.0, 8.5],
                ['112233A', 5.0, 6.0, 8.5, 4.5]]


assert aprobados([['40970455X', 5, 4, 6], ['896737498N', 7, 7,0]]) == ['40970455X']
assert (aprobados(list_notas) == ['33333R', '337464F', '112233A'])

assert alumno_mejor_nota(list_notas) == "33333R" #Mejor nota 9.0

assert (alumno_mejor_nota_media([['40970455X', 5, 4, 10], ['896737498N', 7, 7,10]]) == "896737498N")
assert (alumno_mejor_nota_media(list_notas) == "337464F")

assert (alumnos_peores_notas(list_notas, 2) == ['00000F', '33333R'])
```

(1.5p)

Crea una función que dado una ecuación compruebe si todos los paréntesis, corchetes y llaves se cierran correctamente. No es necesario comprobar que los paréntesis, corchetes y llaves están utilizados de forma correcta en prioridad matemática.

In [ ]:
```python
def comprobar_parentesis_con_texto(texto):
    pass


assert(comprobar_parentesis_con_texto("(-5[{2 + x}])"))
assert(not comprobar_parentesis_con_texto("(3 - 29[2{40+2(}])"))
assert(comprobar_parentesis_con_texto("(x and y or ( not x and [ z or {z and y}]))"))
assert(comprobar_parentesis_con_texto("-3{2x + 5}-(y[xy{3z+4}])"))
assert(not comprobar_parentesis_con_texto("-3}2x + 5{-(y[xy{3z+4}])"))
assert(not comprobar_parentesis_con_texto("{"))
```

(1.5p)

Este ejercicio excepcionalmente no se puede realizar en codereplay ya que no soporte acceso a ficheros, por ello podéis usar cualquier otro IDE. En concreto recomiendo google colab. Podéis ver en un píldora de la semana como trabajar con él.

Implementa la función simulador_servidor_prioridad que simula un servidor que recibe paquetes con diferentes tipos de prioridades.

El simulador puede recibir tres instruciones:

- t *n*. Donde n es el tiempo actual en milisegundos. Actúa como reloj que siempre se incrementa de 10ms en 10ms.
- paquete *c p*. Donde c es el tiempo de caducidad del paquete y p la prioridad. Máxima prioridad es 1.
- perdidos. Retorna el número de paquetes descartados por el servidor hasta ese momento.

El servidor procesa un paquete cada 10 ms (después de recibir el comando t). Si el paquete que le toca ya ha caducado (el tiempo actual es mayor que la caducidad) lo descarta y procesa el siguiente.

In [ ]:
```python
from queue import PriorityQueue
import filecmp


def anadir_paquete(servidor, caducidad, prioridad):
    pass

def procesar(servidor, tiempo):
    pass


def simulador_servidor_prioridad(path_in, path_out):
    servidor          = # Rellenar
    paquetes_perdidos = 0
    tiempo            = 0
    with open(path_in) as f, open(path_out, "w") as f_out:
        lineas = f.readlines() # Leer todas las lineas del fichero
        for l in lineas:
            # Por cada linea del fichero



simulador_servidor_prioridad("test_server_prioridad.txt", "test_server_prioridad_salida.txt")
assert(filecmp.cmp("test_server_prioridad_salida.txt", "test_server_prioridad_salida_correcta.txt"))
#El fichero server_prioridad_salida_correcta.txt esta creado desde una maquina linux, en caso de estar
#  en un sistema windows, este ejercicio ejecutadlo en google colab para testearlo
```

In [ ]: