Evidencia 6 - Trabajo Tournamentsii

Apellidos	García Pascual
Nombre	César
Grupo	1
Comité	Programa
Horas Totales	4 Horas

Evidencias:

1. Creación de problemas para ponerlos en el torneo realizado. Se han realizado 3 problemas en Python. Se adjuntan imágenes tanto del código como de los enunciados inventados para esta actividad.

2. Problemas:

a. Polidivisible

Números Polidivisibles - Mayor Número

Introducción:

En matemáticas se define como número polidivisible a un número natural con las siguientes propiedades:

Sea el número abcde..., definido por sus dígitos, se dice que abcde... es polidivisible si:

Su primer dígito a no es 0.

El número formado por sus dos primeros dígitos ab es múltiplo de 2.

El número formado por sus tres primeros dígitos abc es múltiplo de 3.

El número formado por sus cuatros primeros dígitos abcd es múltiplo de 4.

Etcétera.

Por ejemplo, el número 345654 es un número polidivisible de seis dígitos, pero 123456 no lo es, porque 1234 no es múltiplo de 4.

Problema:

Programar un algoritmo en Java o Python que devuelva el número polidivisible más grande.

```python
#Si un número es polidivisible, al añadir un dígito más al número, si se comprueba
#que el número entero es divisible entre su longitud, este nuevo número será
#polidivisible también.

#Variable que almacenará el resultado
res = 0


##Funcion recursiva que calcula todos los numeros polidivisibles y devuelve el más grande
#Number - Se comprobará si este número es polidivisible. Si lo es, se mirará si es más grande
#que res y si lo es se almacena. Luego se crearan 10 números más a partir de este para
#comprobar si son polidivisibles estos haciendo una llamada recursiva por cada numero.
def getMayorPolidivisible(number):

    #Si el modulo del número entre su longitud es 0
    if (int(number) % len(number) == 0):
        #Si es más grande que res, el numero sustituye a res
        global res
        if (int(number) > int(res)):
            res = number

        #Se hacen 10 llamadas recursivas para el numero actual + 0-9
        for i in range(0, 10):
            getMayorPolidivisible(number + str(i))


##Llamada al algoritmo con los numeros iniciales 1-9
#Para cada número inicial de 1 a 9
for i in range(1, 10):
    #Se comprueba si los derivados de estos son polidivisbles y almacena el mayor
    getMayorPolidivisible(str(i))

#Se muestra el resultado
print(res)
```

b. Desencriptación

Código roto - Cifrado César

Introducción:

En criptografía, el cifrado César, también conocido como cifrado por desplazamiento, código de César o desplazamiento de César, es una de las técnicas de cifrado más simples y más usadas. Es un tipo de cifrado por sustitución en el que una letra en el texto original es reemplazada por otra letra que se encuentra un número fijo de posiciones más adelante en el alfabeto.

Ejemplo:

Aquí se muestra un ejemplo cifrado con un desplazamiento 4:
-Original: ABCD
-Cifrado: EFGH

La A que sería la primera posición (0) si se le desplaza 4 unidades saldría la posición 4, que en el alfabeto actual es la E (A,B,C,D,E,F…)

Aquí otro con desplazamiento 5:
-Original: HOLA
-Cifrado: MTPF

Problema:

Dado el código adjunto en Python y un texto cifrado entregado en un .txt, usar este para descifrar el mensaje (Mensaje en español).
Al encontrar el mensaje cifrado, se ha de entregar un código de 6 letras que aparece al final de este.

```python
#Mensaje y inicializaciones
mensajeCifrado = "u.pdfcr.u qptupvcatcpugpeiupu.pgcvhkqfupscahyuaupuffcfugñpmpsiqahcp qgpgcvhkqfupxqmñp qgpuffcfugpscahyuauopbg,uay"
mensajeClaro = ""

#Caracteres usados en el texto
characters = [
    "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "ñ",
    "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", ",", ".", " "
]

#Variables
desplazamiento = 17

#Algoritmo
for i in range(0, len(mensajeCifrado)):
    mensajeClaro += characters[(characters.index(mensajeCifrado[i]) -
                                desplazamiento) % len(characters)]

#Se muestra por pantalla el mensaje descifrado
print(mensajeClaro)
```

c. Ordenación

Ordenación por decenas

Problema:

Dada una lista de números:

{1, 5, 21, 2, 3, 6, 23, 62, 12, 24, 16, 108, 18, 71, 31, 109, 52, 27, 29,

22, 32, 78, 57, 46, 105, 91, 101, 14, 11, 42, 43, 74, 75, 65, 53, 64, 9,

54, 69, 35, 37, 94, 73, 61, 51, 72, 99, 15, 26, 38, 49, 100, 25, 36, 47, 8}

Ordenar la lista por decenas ascendentemente, y cada decena descendentemente.

Ejemplo:

-Inicial: {28,1,40,24,2,48,15,44,4}

-Ordenada: {4,2,1,15,28,24,48,44,40}

```python
#Lista de números inciales
numeros = [
    1, 5, 21, 2, 3, 6, 23, 62, 12, 24, 16, 108, 18, 71, 31, 109, 52, 27, 29,
    22, 32, 78, 57, 46, 105, 91, 101, 14, 11, 42, 43, 74, 75, 65, 53, 64, 9,
    54, 69, 35, 37, 94, 73, 61, 51, 72, 99, 15, 26, 38, 49, 100, 25, 36, 47, 8
]

print("Lista incial de numeros:", numeros)

#Lista en la que almacenar el resultado
res = []

#Se ordenan los números de menor a mayor
numeros.sort()

#Se saca el mayor
mayor = numeros[len(numeros) - 1]

##Algoritmo que añade a la lista res los numeros por decenas descendentemente
#Para cada decena existente
for i in range(0, mayor // 10 + 1):
    #Primer y ultimo numero de la decena
    primero = int(str(i) + "0")
    ultimo = int(str(i) + "9")
    #Para cada número entre estos se añaden a res los que contiene numeros
    for j in reversed(range(primero, ultimo + 1)):
        if (j in numeros):
            res.append(j)

#Se muestra la lista ordenada
print("Lista ya ordenada:", res)
```