ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN
CCPG1001 - FUNDAMENTOS DE PROGRAMACIÓN
SEGUNDA EVALUACIÓN - I TÉRMINO 2016-2017/ Agosto 30, 2016

Nombre: _____ ***Matrícula:*** _____
Paralelo:____

COMPROMISO DE HONOR: Al firmar este compromiso, reconozco que el presente examen está diseñado para ser resuelto de manera individual, que puedo usar un lápiz o esferográfico; que sólo puedo comunicarme con la persona responsable de la recepción del examen; y, cualquier instrumento de comunicación que hubiere traído, debo apagarlo y depositarlo en la parte anterior del aula, junto con algún otro material que se encuentre acompañándolo. Además, no debo usar calculadora alguna, consultar libros, notas, ni apuntes adicionales a los que se entreguen en esta evaluación. Los temas debo desarrollarlos de manera ordenada. Firmo el presente compromiso, como constancia de haber leído y aceptado la declaración anterior. "Como estudiante de ESPOL me comprometo a combatir la mediocridad y actuar con honestidad, por eso no copio ni dejo copiar".

_____
Firma

TEMA 1 (30 PUNTOS)

Suponga que existe un diccionario **tendencias** con un string que representa una fecha (mm-dd-aaaa) como clave y como valor un conjunto de las etiquetas (hashtags) que fueron tendencias en Twitter para esa fecha. Por ejemplo:

tendencias = {'08-22-2016':{'#Rio2016', '#BSC', '#ECU'}, '08-25-2016':{'#GYE', '#BRA'}, …, '08-27-2016':{'#YoSoyEspol', '#GYE', '#BSC'}}

Implemente las siguientes funciones:

a. **cuentaEtiquetas (tendencias, listaFechas)** que recibe el diccionario de tendencias y una lista con strings que representan fechas (mm-dd-aaaa). La función debe retornar un nuevo diccionario con la etiqueta como clave y como valor, el número de días que esta etiqueta fue tendencia durante las fechas especificadas en *listaFechas*. Por ejemplo:

   cuentaEtiquetas(tendencias, ['08-22-2016', '08-25-2016', '08-27-2016']) retorna

   {'#Rio2106':1, '#GYE':2, '#YoSoyEspol':1, '#BSC':2, '#ECU':1, '#BRA':1}

b. **reportaTendencias(tendencias, listaFechas)** que recibe el diccionario de tendencias y una lista con strings que representan fechas (mm-dd-aaaa). La función debe mostrar por pantalla:
   1) Las etiquetas que fueron tendencia todas las fechas en *listaFechas*
   2) Las etiquetas que fueron tendencia al menos en una de las fechas en *listaFechas*

c. **tendenciasExcluyentes(tendencias, fecha1, fecha2)** que recibe el diccionario de tendencias y dos strings que representan fechas (mm-dd-aaaa). La función debe mostrar por pantalla las etiquetas que fueron tendencias o en fecha1 o en fecha2, pero no en las dos. Nota: suponga que fecha1 y fecha2 existen en el diccionario como claves.

TEMA 2 (60 PUNTOS)

Las distancias en km, expresadas en valores enteros positivos, entre ciudades del Ecuador conectadas directamente por una carretera están dadas en el archivo Ecuador_Distancias.txt con el siguiente formato:

Ciudad de Partida|Ciudad,Distancia|Ciudad,Distancia|...|Ciudad,Distancia

Por ejemplo:

```
Pedernales|Ambato,318|Azogues,555
Ambato|Azogues,280|Babahoyo,212|...|Pedernales,318
Azogues| Pedernales,555|...|Babahoyo,125
        .
        .
        .
Babahoyo|Ambato,250
```

Implemente las siguientes funciones:

a. **cargarDatos(nombreArchivo)** que recibe el nombre del archivo como string y retorna el diccionario **distancias** con el siguiente formato:

{'Ambato':{'Azogues':280, 'Babahoyo':212, … ,'Pedernales':318}, … {'Babahoyo':{'Ambato':250}}}

b. **ciudadesCercanas(distancias, km)** donde **km** es un valor entero positivo y **distancias** es el diccionario generado en el literal a. La función retorna una lista de tuplas con todos los pares de ciudades conectadas directamente por una carretera que estén a una separación menor o igual que **km**. La tupla incluye ciudad1, ciudad2, y distancia que las separa. Por ejemplo:

ciudadesCercanas(distancias, 300) retorna [('Ambato','Azogues',280), ('Ambato','Babahoyo',212), ('Azogues','Babahoyo',125), ('Babahoyo','Ambato',250)]

c. **guardarCiudadesCercanas(distancias, listaKms)** que recibe el diccionario de distancias generado en el literal a y una lista de valores enteros positivos. La función deberá generar por cada valor de la lista un archivo con las ciudades separadas a máximo dicho valor. Por ejemplo:

guardarCiudadesCercanas(distancias, [300, 100, 250]) genera los siguientes tres archivos: ciudades300.txt, ciudades100.txt, ciudades250.txt.

El archivo *ciudades300.txt* tendría el siguiente contenido:

```
Ambato,Azogues,280
Ambato,Babahoyo,212
Azogues,Babahoyo,125
Babahoyo,Ambato,250
```

Escriba un **programa** que lea el archivo Ecuador_Distancias.txt y genere un diccionario de distancias, de acuerdo al formato del literal a. Luego, usando este diccionario, su programa deberá crear los archivos de las ciudades separadas hasta 150, 225, 320 y 555 km.

TEMA 3 (10 PUNTOS)

a. Implemente una función **buscar(listaAnidada, valor)** que recibe una lista de listas y retorna verdadero o falso si *valor* existe en *listaAnidada*. (2.5 puntos)

b. Utilice la función *buscar* del literal a para determinar si el valor X existe en la lista anidada L y mostrar por pantalla 'Valor si existe' o 'Valor no existe'. (2.5 puntos)

```
L = [[3, 2, 5], [1], [7, 9]]
X = int(input('Ingrese valor por teclado')
#su código aquí
```

c. Implemente una función que sume o multiplique valores enteros almacenados en una lista anidada. Si se invoca a la función únicamente con la lista como argumento, la función debe retornar la suma de los valores. Si se invoca a la función con un segundo argumento con valor 'multiplicar', la función debe retornar la multiplicación de los valores. Para cualquier otro valor para el segundo argumento, la función deberá retornar -1. (5 puntos)