Departament de Disseny i Programació de Sistemes Electrònics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Prova Final d'INFORMÀTICA

26/01/2016 Grau en Enginyeria de Sistemes TIC

COGNOMS:

NOM: GRUP de LAB:

Exercici 1 [5 punts]. Coordenades GPS.

Una coneguda empresa vol implantar un aplicatiu que permeti fer amistats properes, tot utilitzant les coordenades GPS on s'ubica cada persona. A tal efecte, ens cal conéixer el funcionament de les coordenades GPS.

Las coordenades GPS estan formades por dues components que són latitud i longitud. Les dues principals unitats de mesura són: 1)coordenades decimals i 2) coordenades sexagesimals.

La latitud i longitud expresada com a 1) coordenades decimals, corresponen a valors decimals. En el cas de la latitud, són valors correctes aquells entre 0..90 i 0..−90. En el cas de la longitud, són valors correctes aquells entre 0..180 i 0..−180. Per exemple, un valor vàlid expressat en coordenades decimals per a una latitud correspondria al valor 42.1361.

En el cas d'expressió de la latitud i longitud com a 2) coordenades sexagesimals, cal tenir en compte que aquestes tenen tres components: graus, minuts i segons. Cadascun d'aquests components sol ser un número enter, però es pot usar un número decimal en els segons si es desitja major precisió. A diferència de les coordenades decimals, les sexagesimals no poden ser negatives. Per exemple, un valor vàlid expressat en sexagesimal per a una latitud correspondria a $42°8'9.96''$. A partir d'ara, ho expressarem com a $42 : 8 : 9.96$

[Apartat a] Per tal de realitzar la conversió de coordenades sexagesimals a coordenades decimals, cal aplicar la següent fórmula:

$graus + (minuts/60) + (segons/3600)$

Per exemple, $42 : 8 : 9.96$ correspon a un format latitud/longitud $42 + (8/60) + (9.96/3600) = 42.1361$, sense pèrdua de decimals.

Per a resoldre els següents apartats, suposeu que totes les dades proporcionades són correctes i no hi ha inconsistències.

Se us demana crear la funció pura *sexagesimalDecimal*, tal que, donat un valor correcte corresponent a latitud/longitud en sexagesimal, retorni la corresponent coordenada decimal. A continuació segueixen els doctests.

```
def sexagesimalDecimal(a):
    """
    >>> sexagesimalDecimal('42:8:9.96')
    42.1361
    >>> sexagesimalDecimal('42:23:60.0')
    42.4
    """
```

[Apartat b] Per tal de realitzar la conversió de coordenades decimals a coordenades sexagesimals, cal aplicar el procediment següent. Prenent el resultat anterior, 42.1361 es converteix de la següent manera,

1. La part no decimal correspon als graus. En aquest cas, 42^o.

2. Multiplicar la part decimal per 60. Els valors han de ser entre el rang 0..23. Exemple: $0.1361 * 60 = 8.166$. Obtenim $8'$. La part decimal pendent és, en aquest cas, 0.166.

3. Multiplicar la part decimal resultant de l'apartat anterior, per 60. Els valors han d'estar entre el rang 0..60. Exemple: $0.166 * 60 = 9.96$. Obtenim $9.96''$.

Se us demana crear la funció pura *decimalSexagesimal* tal que, donat un valor correcte corresponent a latitud/longitud en decimal, retorni la corresponent coordenada sexagesimal. A continuació segueixen els doctests.

```
def decimalSexagesimal(a):
    """
    >>> decimalSexagesimal(42.1361)
    '42:8:9.96'
    >>> decimalSexagesimal(42.4)
    '42:23:60.0'
    """
```

[Apartat c] Per simplificar, suposarem que dues persones estan properes si tenen el mateix valor en graus per la latitud.

Donat un fitxer de text, el nom del qual ve donat per la línia de comandes, tal que conté un llistat de coordenades correctes en format decimal corresponents a la latitud i un identificador-persona, seguint el format següent,

```
42.4 user1
42.1361 user2
34.4 user3
11.2 user4
42.8 user5
34.6 user6
11.8 user7
10.2 user8
```

Es demana que dissenyeu la funció *parellesProperes*, tal que, retorni un llistat de tuples amb les combinacions de parelles properes. Per l'exemple anterior, el resultat hauria de ser,

```
[('user1', 'user2'), ('user1', 'user5'), ('user2', 'user5'), ('user3', 'user6'),
('user4', 'user7')]
```

L'exemple de crida a la funció *parellesProperes* és el que segueix.

```
import sys
if __name__=='__main__':
    if len(sys.argv)==1:
        print 'Manquen dades'
    else:
        print parellesProperes(sys.argv[1])
```

Exercici 2 [3 punts]. Les matrix.

Una matriu dispersa és aquella en què la majoria de valors que conté són zeros. A tal efecte, es decideix emmagatzemar els valors no nuls de dita matriu en un diccionari. Les claus seràn una tupla amb l'indicador de número de fila i número de columna, i el valor associat a dita clau correspondrà a valors no nuls de la matriu. A partir d'ara, d'aquesta representació en direm, representació *dicc-dispersió*. Per exemple, la matriu,

```
0 0 0 1 0
0 0 0 0 3
3 1 0 0 0
0 2 0 0 0
0 0 0 0 1
```

es representaria utilitzant dicc-dispersió com a:

```
m={(0, 3): 1, (1, 4): 3, (2, 0): 3, (2, 1): 1, (3, 1): 2, (4, 4): 1}
```

Per a resoldre els següents apartats, suposeu que totes les dades proporcionades són correctes i no hi ha inconsistències.

[Apartat a] Crear la funció *sumDiagonal*, tal que donada una matriu quadrada en dicc-dispersió, en sumi els valors que composen la diagonal. En aquest cas, suposeu que la matriu que reb la funció sempre serà quadrada. A continuació segueixen els doctests.

```
def sumDiagonal(m):
    """
    >>> sumDiagonal({(0, 3): 1, (1, 4): 3, (2, 0): 3, (2, 1): 1, (3, 1): 2})
    0
    >>> sumDiagonal({(0, 3): 1, (1, 4): 3, (2, 0): 3, (2, 1): 1, (3, 1): 2, (4, 4): 1})
    1
    """
```

[Apartat b] Crear la funció *GuessOrder*, tal que donada una matriu en dicc-dispersió, retorni una tupla (files,columnes) corresponent a l'ordre de la matriu. A continuació segueixen els doctests.

```
def guessOrder(m):
    """
    >>> guessOrder({(0, 3): 1, (1, 4): 3, (2, 0): 3, (2, 1): 1, (3, 1): 2})
    (4, 5)
    >>> guessOrder({(0, 3): 1, (1, 4): 3, (2, 0): 3, (2, 1): 1, (3, 1): 2, (4, 4): 1})
    (5, 5)
    """
```

[Apartat c] Crear la funció *compressRow*, tal que donada una matriu en dicc-dispersió, retorni el resultat de comprimir la informació de la matriu utilitzant el següent mètode:

1. s'obté la mitjana dels elements per fila
2. cal emmagatzemar l'indicador de la fila, el nombre de columnes per fila amb valors no nuls i el valor de la mitjana anterior

Per exemple, la matriu en dicc-dispersió

```
m={(0, 3): 1, (1, 4): 3, (2, 0): 4, (2, 1): 1, (3, 1): 2, (4, 4): 1}
```

quedaria comprimida com a:

```
m={(0, 1): 1.0, (1, 1): 3.0, (2, 2): 2.5, (3, 1): 2.0, (4, 1): 1.0}
```

A continuació segueixen els doctests de la funció *compressRow*.

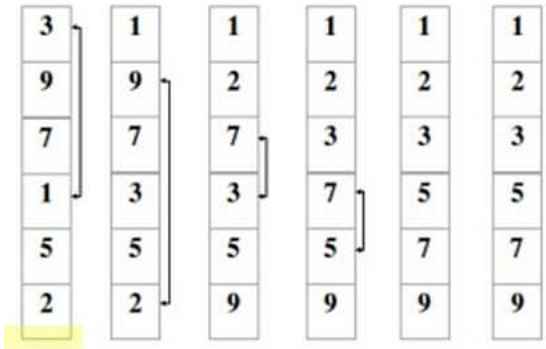
```
def compressRow(m):
    """
    >>> compressRow({(0, 3): 1, (1, 4): 3, (2, 0): 3, (2, 1): 2, (3, 1): 2})
    {(0, 1): 1.0, (3, 1): 2.0, (1, 1): 3.0, (2, 2): 2.5}
    >>> compressRow({(0, 0): 1, (0, 3): 2, (1, 0): 2, (1, 1): 1, (1, 3): 1, (2, 3): 1})
    {(1, 3): 1.3333333333333333, (0, 2): 1.5, (2, 1): 1.0}
```

Nota: La no utilització del format especificat en l'enunciat per la gestió de les matrius, comportarà una puntuació nul·la.

Exercici 3 [2 punts]. La llista ordenada.

Una de les estratègies utilitzades en l'ordenació d'elements, més senzilles d'implementar, és la *ordenació per selecció*. El funcionament d'aquesta ordenació consisteix en buscar, per a cada iteració **i**, quin és l'**i**-èssim número més petit. Quan l'ha trobat l'intercanvia amb l'element que, en aquell moment, està a la posició **i** de la llista que s'està ordenant.

A continuació s'exemplifica gràficament el seu funcionament.



Se us demana que implementeu la funció **pura** *selectionSort* tal que, donada una llista de números correctes, retorni la llista ordenada ascendentment utilitzant l'estratègia *ordenació per selecció*.

Òbviament, la utilització de les funcions predefinides de Python en l'ordenació, suposarà una puntuació nul·la.