UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Departament d'EMIT-UPC-EPSEM

Prova Final d'INFORMÀTICA

26/01/2017 Grau en Enginyeria de Sistemes TIC

COGNOMS:

NOM: GRUP de LAB:

Exercici 1 [5 punts]. Acrònim-encriptació de missatges. Com ja coneixeu, una de les formes d'encriptació de paraules és el xifratge Cèsar. A continuació se us demana que implementeu una altra estratègia de resolució, que anomenarem **acrònim-encriptació**. Per tal de codificar un missatge amb la tècnica *acrònim-encriptació*, simplement cal agafar la primera lletra de cada paraula, obtenint així un nou missatge. Per exemple, donat el missatge *"les vacances de primer quadrimestre estan a punt"*, el resultat de l'acrònim-encriptació seria el missatge *"lvdpqeap"*. Per tal de resoldre els apartats que segueixen, podeu suposar que els missatges no contindran lletres majúscules.

[**Apartat 1.a**] Se us demana que realitzeu la funció pura *encriptaAcronimous*, tal que, donat un missatge, retorni el missatge codificat segons la tècnica acrònim-encriptació. Suposeu que el separador de paraules dins el missatge és un espai en blanc, i que el missatge únicament contindrà lletres alfabètiques.

```
def encriptaAcronimous(s):
    """
    >>> encriptaAcronimous("les vacances de primer quadrimestre estan a punt")
    'lvdpqeap'
    """
```

[**Apartat 1.b**] En aquest apartat se us demana implementar la funció *decodificaAcronimous*, encarregada de decodificar un missatge secret encriptat amb *acrònim-encriptació* seguint l'estratègia que es detalla a continuació.

Donat un missatge secret, *decodificaAcronimous* construeix un nou missatge, on cada lletra del missatge original es canvia per una paraula que comenci per aquesta lletra. En cas que el caràcter no sigui alfabètic, aquest es manté. Finalment, la funció ha de retornar aquest nou missatge. Per exemple, *decodificaAcronimous* amb el missatge "nj, nr", pot retornar "nova jornada, nous reptes".

Per implementar la funció, utilitzeu la funció que se us proporciona resolta, de nom **obtenirParaula**, tal que, donat un caràcter, retorna una paraula aleatòria que comenci per aquest caràcter.

```
def decodificaAcronimous(s):
    """
    >>> decodificaAcronimous("nj, nr")
    'nova jornada, nous reptes'
    """
```

[**Apartat 1.c**] En aquest cas se us demana gestionar la funcionalitat que permet obtenir una paraula aleatòria que comenci per una lletra. En aquest cas se us demana **1.c.1)** un script que llegeixi prèviament un fitxer de text que contingui un conjunt de paraules, i emmagatzemi òptimament cada paraula en una estructura. Aquesta estructura ha de ser òptima de manera que donada una lletra, obtingui la llista de paraules del fitxer que comencin per aquesta lletra. Suposeu que el fitxer de paraules existeix amb el nom *paraules.txt* i que cada paraula en el fitxer està separada d'una altra per un espai en blanc. Tingueu en compte que les paraules del fitxer no segueixen cap ordre. I que una mateixa paraula pot estar repetida en el fitxer vàries vegades.

Finalment, us cal **1.c.2**) implementar la funció *obtenirParaula*, tal que utilitzi la estructura de dades anterior per, donat un caràcter, retornar una paraula aleatòria de la llista que comenci per aquest caràcter.

Se us demana que el codi Python dels passos c.1) i c.2) amb el funcionament esperat.

Exercici 2 [3 punts]. Sudokus. Sudoku és un joc matemàtic l'objectiu del qual és omplir una quadrícula de 9x9 cel.les (81 caselles) dividida en subquadrícules de 3x3 (també anomenades "caixes" o "region") amb les xifres de l'1 al 9 partint d'alguns nombres ja disposats en algunes de les cel·les, tal i com es mostra a la imatge següent.

2			3					
8		4		6	2			3
	1	3	8			2		
				2		3	9	
5		7				6	2	1
	3	2			6			
	2				9	1	4	
6		1	2	5		8		9
					1			2

L'objectiu del joc és omplir cadascun dels espais lliures amb dígits entre l'1 i el 9 de manera que cada dígit aparegui exactament un cop en cada fila, en cada columna, i en cada "caixa". Cada puzzle Sudoku es construeix amb cura donat que només hi ha una única sol·lució. Un exemple correcte de sudoku és el que segueix.

2	7	6	3	1	4	9	5	8
8	5	4	9	6	2	7	1	3
9	1	3	8	7	5	2	6	4
4	6	8	1	2	7	3	9	5
5	9	7	4	3	8	6	2	1
1	3	2	5	9	6	4	8	7
3	2	5	7	8	9	1	4	6
6	4	1	2	5	3	8	7	9
7	8	9	6	4	1	5	3	2

Suposeu que se us demana la creació d'un script que comprovi si una sol·lució proposada és correcta. Donat que la tasca és excessivament complexa per a resoldre en un problema d'examen, se us demana una simplificació, que consisteix en comprovar si una "caixa" determinada conté els dígits de l'1 al 9. En l'exemple anterior, qualsevol de les "caixes" 3x3 ho és.

Si per exemple, l'usuari ha comés un error en una caixa, omplint el puzzle tal com segueix, la solució serà invàlida donat que conté dues vegades el valor 5 i el valor 6 no hi apareix cap cop.

2	7	5
8	5	4
9	1	3

[**Apartat 2.a**] Dissenyeu **òptimament** la funció *chequejaCaixa*, tal que donada una matriu 9x9 inicialitzada amb valors enters, retorni **True** únicament si la **caixa superior esquerra** és correcta. És a dir, conté els dígits de l'1 al 9. Si la caixa conté un enter fora del rang permès o bé conté valors duplicats, la funció ha de retornar **False**. En canvi, en l'exemple que segueix,

2	7	6
8	5	4
9	1	3

la funció retornaria **True**.

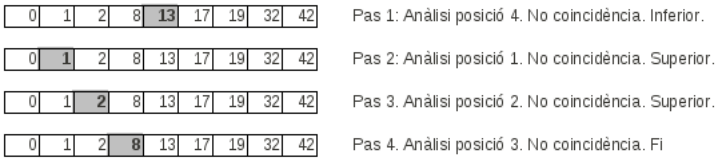
```
def chequejaCaixa(m):
    """
    >>> chequejaCaixa([[2,7,6,3,1,4,9,5,8],[8,5,4,9,6,2,7,1,3],[9,1,3,8,7,5,2,6,4],
[4,6,8,1,2,7,3,9,5],[5,9,7,4,3,8,6,2,1],[1,3,2,5,9,6,4,8,7],
[3,2,5,7,8,9,1,4,6],[6,4,1,2,5,3,8,7,9],[7,8,9,6,4,1,5,3,2]])
    True
    >>> chequejaCaixa([[2,7,5,3,1,4,9,6,8],[8,5,4,9,6,2,7,1,3],[9,1,3,8,7,5,2,6,4],
[4,6,8,1,2,7,3,9,5],[5,9,7,4,3,8,6,2,1],[1,3,2,5,9,6,4,8,7],
[3,2,5,7,8,9,1,4,6],[6,4,1,2,5,3,8,7,9],[7,8,9,6,4,1,5,3,2]])
    False
    """
```

[**Apartat 2.b**] Justifiqueu què us caldria modificar en la funció anterior per tal que permetés comprovar qualsevol de les caixes. (No cal el codi Python, només una justificació raonada dels canvis).

Exercici 3 [2 punts]. Cercant un element. Una de les estratègies utilitzades per comprovar si un element es troba emmagatzemat en una llista és la *cerca binària*. Suposem una llista ordenada ascendentment. El funcionament consisteix en localitzar l'element del centre de la llista i comprovar si és el que busquem. Si aquest no coincideix amb l'element que es cerca, es determina en quina meitat de la llista ha de ser-hi. Es cerca en aquesta meitat repetint el procés les vegades que calgui fins trobar l'element, o bé fins que es determini que no es troba a la llista. Fixeu-vos que es tracta d'una estratègia divideix-i-venceràs, donat que en cada pas es minimitza la llargada de la llista a comprovar.

A continuació s'exemplifica gràficament el seu funcionament, suposant que:

la llista és: $[0, 1, 2, 8, 13, 17, 19, 32, 42]$, el valor a cercar correspon al 3.



Se us demana que implementeu la funció **pura** *binarySearch* tal que, donada una llista d'elements ordenada ascendentment, i un element a cercar, retorni **True** si l'element es troba a la llista i **False** en cas contrari, utilitzant l'estratègia *cerca binària*.

Òbviament, la utilització de les funcions predefinides de Python (excepte la funció *len*) en la cerca, suposarà una puntuació nul·la.