Algorithms and Python Programming
Assignment 1

recursions, dictionaries

by Hadas Lapid

1.A.

Implement the **recursive** function **'countCInList'**.

The function receives a list L and an integer, c. the function returns the number of appearances of c in L. If c does not appear in L it returns 0.

For example, given the list: l = [1, 4, 7, 1, 2, 1]

When c=1 the output is 3

When c=7 the output is 1

When c=8 the output is 0

When c=2 the output is 1

(see function tests in 'main')

1.B.

Implement the function **'isCNInList'**. The function receives a list, L, an integer c, and an integer, n, who's default value is 0.

The function returns a Boolean, indicating whether c appears in L n times. For this, the function uses 'countCInList' subroutine from question 1.A.

For example, given the list: l = [1, 4, 7, 1, 2, 1]

When c=1 and n=3 the function returns True

When c=1 and n=2 the function returns False

When c=7 and n=1 the function returns True

When c=8 and n=3 the function returns False

When c=8 and n is not provided, the function returns True

(see function tests in 'main')

2.

Implement a **recursive** function **‘countNumsDict’**.

The function gets a list of elements, l, and a default dictionary, d.

The function returns a dictionary whose keys are the unique elements in l, and their corresponding values are their respective number of appearances.

For example, for the list l = [1, 4, 7, 1, 2, 1], the function returns the dictionary:

{1: 3, 4: 1, 7: 1, 2: 1}

For the sentence "Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world", cut into words in lower case, the function returns the dictionary:

{'imagination': 2, 'is': 2, 'more': 1, 'important': 1, 'than': 1, 'knowledge.': 1, 'knowledge': 1, 'limited.': 1, 'encircles': 1, 'the': 1, 'world': 1}

3.

Implement the **recursive** function **‘int2bin’**.

The function gets an integer, n, and a default integer, i, and returns the binary value of n (as int).

Hint: the function uses the ‘i’ for the binary calculation.

Notice: Only recursive solutions will be accepted.

For example, when evoking int2bin with 175, it returns 10101111.

when evoking int2bin with 17, it returns 10001

when evoking int2bin with 7, it returns 111

when evoking int2bin with 235, it returns 11101011

4.

Implement the **recursive** function **recDiag**.

The function gets a rectangle list of characters, l, and a default integer, i, representing list’s index. The function returns a Boolean indicating if the list’s diagonal elements are in sequential order.

Hint: i is not required for initial function evocation, only for recursive purposes.

Hint2: use ord() function to get the numerical value of a characters.

Notice: Only recursive solutions will be accepted.

For example, given the list l1:

```
l1 =[['a','g','d','y','b'],
    ['n','b','u','k','t'],
    ['l','t','c','p','i'],
    ['y','f','v','d','e'],
    ['e','r','b','r','e']]
```

recDiag(l1) returns True

given the list l2:

```
l2 = [['d', 'g', 'w', 'y', 'a'],
     ['n', 'c', 'u', 'k', 'j'],
     ['l', 's', 'b', 'p', 'l'],
     ['k', 'f', 'r', 'm', 'e'],
     ['q', 'r', 'b', 'r', 'n']]
```

recDiag(l1) returns False

Success = 90% Persistence and 10% talent

Good Luck!

Hadas