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

EXERCICI PUNTUABLE TECPRO

29/05/2017 Grau en Enginyeria de Sistemes TIC

COGNOMS:

NOM: GRUP de LAB:

Exercici 1. Donada les següents funcions recursives, a) Justifiqueu el resultat de cadascun dels seus doctestos b) Pel *doctest2* i *doctest5*, justifiqueu quantes crides recursives s'efecutaran c) Digueu quin és el cost assimptòtic en cas pitjor de dita funció.

```
Apartat a)
def f(L):
    """
    >>> f('3')
    #doctest1
    >>> f([1, [[2, 'a'], ['a','b']], (3, 4)])
    #doctest2
    >>> f(3)
    #doctest3
    """
    result = []
    for e in L:
        if type(e) != list:
            result.append(e)
        else:
            return f(e)
    return result
```

```
Apartat b)
def f(s):
    """
    >>> f('electrode')
    #doctest4
    >>> f('sensor')
    #doctest5
    """
    if len(s) <= 1:
        return s
    return f(f(s[1:])) + s[0]
```

Exercici 2. Donada una expressió correcta, se us demana que comproveu si conté parèntesis duplicats o no. Un conjunt de parèntesis són duplicats si la mateixa subexpressió està rodejada de múltiples parèntesis. A continuació es mostren expressions amb parèntesis duplicats.

```
#Exemple 1
((a+b)+((c+d))) #La subexpressió ''c+d'' està envoltada de 2 parells de parèntesis.

#Exemple 2
(((a+(b)))+(c+d)) #La subexpressió ''a+(b)'' està envoltada per dos parells de parèntesis.

#Exemple 3
(((a+(b))+c+d)) #L'expressió completa està envoltada per dos parells de parèntesis.

Les expressions que segueixen a continuació no contenen parèntesis duplicats.

#Exemple 4
((a+b)+(c+d)) #Cap subexpressió està rodejada de parèntesis duplicats.

#Exemple 5
((a+(b))+(c+d)) #Cap subexpressió està rodejada de parèntesis duplicats.
```

Se us demana que implementeu la funció booleana *findDuplicate(expression)*, tal que, donada una expressió correcta retorni True si conté parèntesis duplicats.

Nota: podeu utilitzar la implementació de l'estructura de dades lineal pila (Stack), i les seves operacions push(), top(), pop(), len (no cal que lliureu la implementació de la classe Stack, podeu utilitzar-la directament).

Exercici 3. Donada la llista següent desordenada,

```
[54, 26, 93, 17, 77, 31, 44, 55, 20]
```

Apartat a) Detalla com s'ordenaria ascendentment pas per pas aplicant el mecanisme d'ordenació *bubble sort*. Quina és la complexitat en cas pitjor del bubble sort?

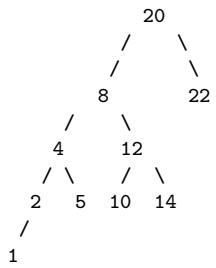
Apartat b) Detalla com s'ordenaria ascendentment pas per pas aplicant el mecanisme d'ordenació *merge sort*. Quina és la complexitat en cas pitjor del merge sort?

Exercici 4. Donada la classe —arbreBST—, corresponent a un Binary Search Tree, amb els mètodes implementats a classe de teoria, com el que segueix.

```
class arbreBST(object):
    def __init__(self):
        self.v=None
        self.left=None
        self.right=None

    def insereix(self,k):
        if self.v==None:
            self.v=k
        else:
            if self.v==k:
                raise Exception("Repetit")
            elif self.v<k:
                if self.right is None:
                    self.right=arbreBinari()
                self.right.insereix(k)
            else:
                if self.left is None:
                    self.left=arbreBinari()
                self.left.insereix(k)
```

Apartat a) Se us demana que implementeu el mètode recursiu *minValue*, tal que, donat un arbre BST no balancejat, retorni el número més petit emmagatzemat a l'arbre. Per exemple, en l'arbre següent, la resposta hauria de ser 1.



Apartat b) Escriu el mètode recursiu *printArbrePreordre*, tal que, donat un arbre com l'anterior escrigui els nodes seguint un recorregut de l'arbre en preordre.

Per l'arbre de la figura, el seu recorregut en preordre hauria de ser:

20 8 4 2 1 5 12 10 14 22

Apartat c) Escriu quina és la complexitat en cas pitjor del mètode *minValue* i quina és la complexitat en cas pitjor del mètode *printArbrePreordre*. Utilitza la notació O, i justifica breument el seu cost.