EXERCICI PUNTUABLE TECPRO

21/05/2018 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

```
Apartat a)
def mistery(s):
    """
    >>> mistery('program')
    #doctest1
    >>> mistery('python')
    #doctest2
    """
    if len(s) <= 1:
        return s
    return mistery(mistery(s[1:])) + s[1]
```

```
Apartat b)
def noIdea(L):
    """
    >>> noIdea('88')
    #doctest3
    >>> noIdea([11, [[22, 'ab'], ['ba','bx']], (31, 41)])
    #doctest4
    >>> noIdea(44)
    #doctest5
    """
    result = []
    for e in L:
        if type(e) != list:
            result.append(e)
        else:
            return noIdea(e)
    return result
```

Exercici 2. Justifica si són correctes o falses les afirmacions que segueixen. Una resposta sense justificació no serà avaluada.

1. La funció append de Python aplicada a una llista d'elements, té una complexitat en cas pitjor de $O(n)$.
2. L'operació '+' de Python aplicada sobre llistes, té una complexitat en cas pitjor de $O(n)$, sent n el nombre d'elements de la llista més gran a concatenar.
3. La cerca binària d'un element en una llista de n elements té una complexitat en cas pitjor de $O(n)$.
4. La cerca en un arbre binari de cerca de n nodes, d'un element que no es troba en l'arbre, tindria un cost de $O(n)$.

Exercici 3. Donada la llista següent desordenada,

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

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

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

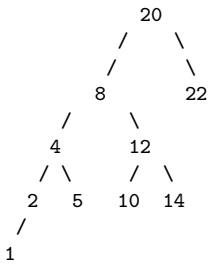
Exercici 4. Donada la classe —arbreBST—, corresponent a un Binary Search Tree, amb els mètodes implementats com segueixen,

```
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=arbreBST()
                self.right.insereix(k)
            else:
                if self.left is None:
                    self.left=arbreBST()
                self.left.insereix(k)
    def maxValue(self):
        #TO DO
    def printArbrePostordre(self):
        #TO DO
```

```
if __name__=='__main__':
    arbre=arbreBST()
    arbre.insereix(20)
    arbre.insereix(8)
    arbre.insereix(12)
    arbre.insereix(4)
    arbre.insereix(10)
    arbre.insereix(14)
    arbre.insereix(22)
    arbre.insereix(2)
    arbre.insereix(5)
    arbre.insereix(1)
    print arbre.maxValue()
    arbre.printArbrePostordre()
```

Apartat a) Se us demana que implementeu el mètode recursiu *maxValue*, tal que, donat un arbre BST no balancejat, retorni el número més gran emmagatzemat a l'arbre. Per exemple, en l'arbre següent, la resposta hauria de ser 22.



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

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

1 2 5 4 10 14 12 8 22 20

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