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

Prova final de TECPRO

10/06/2014 Grau en Enginyeria de Sistemes TIC

COGNOMS:

NOM: GRUP de LAB:

Exercici 1. Defineix breument i posa un exemple on es vegin clarament reflectits els conceptes que segueixen.

1. Ocultació d'informació.
2. Sobrecàrrega de mètodes.
3. Delegació de mètodes.
4. Interrelació de composició.

Exercici 2. Escriu quin serà els resultat del següent fragment de codi

Apartat a)

```
class NotWorking(Exception):
    pass
def comparison(a,b):
    try:
        if a>b:
            raise NotWorking('Bigger')
        else:
            return 15*a
    except Exception:
        return -1

if __name__=='__main__':
    try:
        print comparison(comparison(4,8),2)
    except Exception:
        print "not yet"
```

Apartat b)

```
from datetime import datetime
dt = datetime.strptime("10/06/14 10:00", "%d/%m/%y %H:%M")
print dt.strftime("%A, %d. %B %Y %I:%M%p")
```

Exercici 3. Escriu quin serà el resultat de cadascuna de les següents expressions.

```
>>> 32>>5
#TO DO
>>> 166&2**5
#TO DO
>>> 166&1>>0
#TO DO
>>> 4|1<<4
# TO DO
>>> 4&~(1<<2)
#TO DO
```

Exercici 4. Dibuixa i justifica el diagrama UML que permetria modelar el següent problema,

Apartat a) Es preten emmagatzemar les dades referents a diversos sensors de temperatura. Cada sensor es classifica en sensors impermeables o sensors no impermeables. Les dades referents a les temperatures es recullen en un històric anual, d'on se'n guarda el dia, la franja horària (format hh:mm:ss) i el sensor que ha recollit la dada.

Apartat b) Justifica com s'implementaria en Python (breu descripció dels atributs / mètodes que les classes contindran).

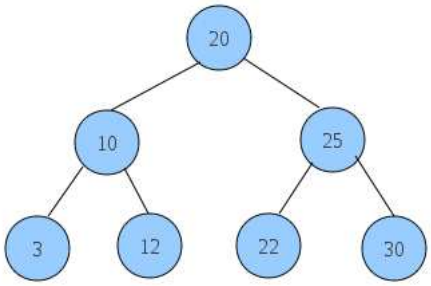
Exercici 5. Afegiu a la definició d'arbre binari de cerca que segueix, els mètodes recursius que se us detallen a continuació,

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

    def maxim(self):
        """
        Mètode recursiu que retorna el valor mes gran de l'arbre
        """
        #TO DO

    def interval(self,inici,final):
        """
        Mètode recursiu que mostra els elements de l'arbre en preordre i que es trobin en l'interval (inici,final)
        """
        #TO DO
```

Per exemple, donat el següent arbre binari de cerca,



1. el resultat de la funció maxim hauria de ser: 30
2. el resultat de la funció interval(4,27) hauria de ser: 20, 10, 12, 25, 22

Exercici 6. L'expressió correcta.

Suposem que cal gestionar la correcta parentització d'una expressió en un sistema d'enviament d'informació. Hi ha tres tipus d'elements que considerem parèntesi: (), [] i {}. Suposeu que no s'envia res més que no siguin aquests parèntesi d'obertura/tancament i que l'ordre de parèntesi és important. Per exemple, {()} està ben parentitzada, però {(}) no.

1) Escriviu una funció tal que donada una expressió d'aquest tipus, retorni **True** o bé **False** si aquesta està ben parentitzada. Comproveu els doctests que figuren a continuació.

Pista: Podeu utilitzar la classe Stack directament i els seus mètodes associats.

2) Justifiqueu quina complexitat en cas pitjor té la funció que heu dissenyat.

```
def benParentitzada(e):
    """
    >>> benParentitzada('([])')
    True
    >>> benParentitzada('(((((((([[[{{}]]])))))))')
    True
    >>> benParentitzada('{[(]})')
    False
    """
```

Exercici 7. La codificació Run-length.

Run-length encoding, RLE, és una forma molt simple de compressió de dades en què seqüències de dades amb el mateix valor consecutiu són emmagatzemades com un únic valor més el seu recompte. Per exemple, si es considera una pantalla que conté text en negre sobre un fons blanc. Hi ha moltes seqüències d'aquest tipus amb píxels blancs en els marges buits, i altres seqüències de píxels negres a la zona del text. Suposant una línia (o scanline), on N representa les zones en negre i B les de blanc:

BBBBBBBBBBBBNBBBBBBBBBBBBNNNBBBBBBBBBBBBBBBBBBBBBBNBBBBBBBBBBBBB

Si s'aplica la codificació run-length a aquesta línia, s'obtindrà el següent:

12B1N12B3N24B1N14B

Fixeu-vos que en la seqüència resultant, la primera dada correspon al nombre de vegades que el caràcter que ve a continuació està repetit.

L'objectiu del problema és dissenyar una classe de nom RLE que permeti representar i manipular seqüències RLE. En concret, donat el següent esquelet de la classe RLE, es demana implementar els mètodes *encode* i *decode* i els mètodes necessaris per tal que el funcionament del joc de proves sigui l'esperat.

```
class RLE(object):

    def __init__(self,seq):
        self.releSeq=seq
        self.encode()

    def encode(self):
        """
        saves the RLE-encoded string
        """
        #TO DO

    def decode(self):
        """
        returns the string corresponding to the RLE-encoded string for the class instance
        """
        #TO DO

if __name__=='__main__':
    a='BBBBBBBBBBBBNBBBBBBBBBB'
    r=RLE(a)
    print "codificacio", r
    print "decodificacio", r.decode()
```

El resultat de l'execució és el que segueix.

```
codificacio 12B1N11B
decodificacio BBBBBBBBBBBBNBBBBBBBBBB
```

Exercici 8. El Referèndum.

L'objectiu del problema a resoldre consisteix a desenvolupar un sistema que gestioni les votacions de candidats en diferents regions. Suposant la següent definició de la classe Referendum, dissenyeu els mètodes que manquen i els que permetin el correcte funcionament del joc de proves que segueix.

```
class Referendum(object):
    def __init__(self):
        self.regions=['r1','r2','r3']
        self.candidats=['A','B','C']
        self.votsRegions={}

    def afegirVots(self,c,v,r):
        if not r in self.votsRegions:
            self.votsRegions[r]={c:v}
        else:
            if not c in self.votsRegions[r]:
                self.votsRegions[r][c]=v
            else:
                self.votsRegions[r][c]+=v

    def votsCandidatRegions(self,c):
        """
        retorna el nombre total de vots pel candidat c, en totes les regions
        """
        #TO DO

    def mesVotatRegio(self,r):
        """
        retorna el candidat mes votat d'una regio
        """
        #TO DO

    def llistaRegions(self,c):
        """
        retorna un llistat amb les regions on ha guanyat el candidat
        """
        #TO DO

if __name__=='__main__':
    v=Referendum()
    v.afegirVots('A',100,'r1')
    v.afegirVots('A',55,'r2')
    v.afegirVots('A',150,'r3')
    v.afegirVots('C',200,'r1')
    v.afegirVots('B',50,'r2')
    v.afegirVots('C',120,'r3')
    print v
    print v.votsCandidatRegions('A')
    print v.mesVotatRegio('r3')
    print v.llistaRegions('A')
```

El resultat de l'execució és el que segueix.

```
r1 {'A': 100, 'C': 200}
r2 {'A': 55, 'B': 50}
r3 {'A': 150, 'C': 120}

305
A
['r2', 'r3']
```