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

Prova final de TECPRO

17/06/2016 Grau en Enginyeria de Sistemes TIC 2H

COGNOMS:

NOM: GRUP de LAB:

Llegiu atentament els enunciats i responeu les preguntes que segueixen. Lliureu les respostes a cada exercici en fulls separats.

Exercici 1 [2 punts]. Preguntes Cert/Fals. Per cadascun dels enunciats següents, escriu si és [Cert/Fals]. En cas que sigui **Fals, justifica la resposta**. (La no justificació invalida la resposta)

1. Els algorismes selection sort, bubble sort i heap sort tenen el mateix cost assimptòtic en cas pitjor.
2. La cerca binària requereix cost linial en cas pitjor.
3. Una pila és una estructura FIFO, i una cua és una estructura LIFO.
4. Sempre hi haurà un cost en temps $O(logn)$ per cercar un node arbitrari en un arbre de cerca binari de tamany n.
5. Una subclasse pot accedir a mètodes privats de la superclasse.
6. Un mètode que triga $O(logn)$ calcularà un resultat més ràpid que un mètode que triga $O(n)$, assumint que n té el mateix valor per cada mètode.
7. Merge sort és un algorisme amb una complexitat en cas pitjor de $O(nlogn)$.
8. Suposem que B és una subclasse de A i C és una subclasse de B. Llavors, tot mètode d'un objecte de classe C pot executar el codi del constructor de A.
9. El block finally en un try..except..else..finally s'executa inclús s'hi no s'ha llençat una excepció i hi ha una instrucció return en el block try..except.
10. Si una classe A té dos mètodes amb signatures $m(self, m = "")$ i $m(self, v = 0)$, diem que el mètode m està redefinit.

Exercici 2 [3 punts]. Preguntes de resposta curta.

Apartat a) Una clínica dental demana assessorament TIC per tal de portar un sistema de registre dels seus pacients. A continuació segueix el llistat de requeriments que han fet arribar.

Per cada pacient cal emmagatzemar el seu nom, adreça i un número de mòbil. A cada pacient se li assigna un número únic de set dígits. El sistema necessita portar un recompte dels pacients que té en tot moment. El pacients poden demanar una visita amb un dentista en concret; en tal cas, el sistema ha d'emmagatzemar la data de la visita i si el pacient va ser atès o no. S'enviarà al pacient un missatge automàtic de recordatori 2 dies previs a la visita. Un cop finalitzada la visita, el dentista ha de modificar el sistema afegint el cost del tractament.

Hi ha dos tipus d'empleats: els recepcionistes i els dentistes. D'ambdòs, el sistema ha d'emmagatzemar les seves particularitats. Per tots els empleats, cal un número d'empleat de quatre dígits, el nom, adreça, gènere, número de telèfon i el parent més proper. Els dentistes estan qualificats, per tant el sistema ha de guardar la seva qualificació més alta, la data en que es va obtenir i el número de registre del Col·legi de Dentistes.

Al final de cada setmana cal generar una llista estadístiques de visites. Aquest llistat ha de ser un resum de quans pacients es van presentar i quants no es van presentar. Si el pacient no es presenta a una visita, se li carregarà una quantitat fixa de diners.

Tots els recepcionistes han de realitzar anualment un curs de primers auxilis. el sistema ha de gestionar la data de l'últim cop que van realitzar el curs i el nom del professor que va impartir el curs.

Se us demana que realitzeu el diagrama de classes que permeti resoldre el problema plantejat, amb especificació dels atributs i la signatura dels principals mètodes. No calen mètodes constructors ni accessors/modificadors. Únicament cal la signatura, no la implementació.

Apartat b) Implementa la classe esquiador. Tot esquiador ha de tenir un nom (públic), un identificador (privat), i un llistat de puntuacions obtingudes. Cal que en gestioneu el constructor i els mètodes necessaris per tal de tenir una definició de classe completa.

Apartat c) Prenent com a base la classe desenvolupada en l'Apartat b), escriu un exemple de 1) mètode sobrecarregat, 2) mètode redefinit, 3) herència de mètodes i 4) delegació de mètodes.

Apartat d) Escriu els resultats que es mostraran per pantalla.

```
class A(object):
    def first(self,x):
        try:
            print "First in"
            self.second(x)
        except ZeroDivisionError as e:
            print "Math"
        except Exception as e:
            print "Firs Except"
        print "First out"

    def second(self,x):
        print "Second in"
        if x<0: raise Exception()
        y=5/x
        print "second out"
if __name__=='__main__':
    a=A()
    a.first(0)
    a.first(ord("A"))
```

Exercici 3 [3 punts]. Implementació de classes.

Has acceptat treballar a Facebook-EPSEM i estàs treballant en una nova app de nom "TotSobreMi". L'objectiu de dita aplicació és mostrar-te els comentaris de les fotos de tu mateix que han pujat altres persones. Hauràs de resoldre el problema en tres passos/apartats. L'apartat b) usa el mètode de l'apartat a), i l'apartat c) usa el b). A continuació et passen el prototipus de classes per Facebook-EPSEM per gestionar la informació i un exemple de funcionament. Cada Usuari té un nom, un conjunt d'amics i un conjunt de fotografies. Cada foto té una imatge i un conjunt de tags. Aquests tags poden incloure noms d'usuari (informació rellevant per la nova app).

```
class User(object):
    def __init__(self,nom):
        self.name=nom
        self.friends=[]
        self.photos=[]

    def showPhotos(self):
        print "Photos user",self.name
        for p in self.photos:
            print "Name of image",p.image,"list of tags"
            for t in p.tags:
                print t
```

```
    def addPhoto(self,p):
        if p not in self.photos:
            self.photos.append(p)

class Photo(object):
    def __init__(self,image):
        self.image=image
        self.tags=[]

    def addTag(self,t):
        self.tags.append(t)

    def __eq__(self,other):
        return self.image==other.image

if __name__==’__main__’:
    g=User("pere")
    p=Photo("a.jpg")
    p1=Photo("b.jpg")
    g.addPhoto(p)
    g.addPhoto(p1)
    p.addTag("pere Fantastic")
    p1.addTag("wow pere")
    p1.addTag("quants anys...")
    g.showPhotos()

#Resultat execució
Photos user pere
Name of image a.jpg list of tags
pere Fantastic
Name of image b.jpg list of tags
wow pere
quants anys...
```

Apartat a) Implementar el mètode $myTags(self)$ a la classe User, tal que ha de retornar una llista dels tags tals que contenen el name de l'user en les photos d'aquest user. Seguint l'exemple anterior, a continuació segueix quin hauria de ser el resultat d'invocació del mètode.

```
print g.myTags()

#Resultat d’execució
[’pere Fantastic’, ’wow pere’]
```

Apartat b) Implementar el mètode $tagsAmics(self)$ a la classe User, tal que retorni un llistat de tags tals que contenen el nom de l'user en els tags de les seves fotos, juntament amb aquells tags on apareix el seu nom en el llistat de fotos dels seus amics. Cal utilitzar el mètode definit en l'Apartat a), i si us cal, podeu canviar-ne la signatura del mètode. A continuació segueix exemple d'execució.

```
a=User("maria")
t=Photo("c.jpg")
a.addPhoto(t)
t.addTag("holidays with pere")
g.addFriend(a)
print g.tagsAmics()

#Resultat d’execució
[’pere Fantastic’, ’wow pere’, [’holidays with pere’]]
```

Apartat c) Implementar el mètode $totSobreMi(self)$ a la classe User, tal que retorni un llistat de tags tals que contenen el nom de l'user en els tags de les seves fotos, juntament amb aquells tags on apareix el seu nom en el llistat de fotos dels seus amics, i amics d'amics. Seguint amb la seqüència d'instruccions anterior i la que segueix, cal obtenir el resultat que es mostra a continuació.

```
c=User("joanC")
f=Photo("d.jpg")
c.addPhoto(f)
f.addTag("holidays with maria and pere")
a.addFriend(c)
r=User("esteve")
s=Photo("e.jpg")
r.addPhoto(s)
s.addTag("holidays with maria and pere and joan")
s.addTag("pere always smiling")
c.addFriend(r)
print g.totSobreMi()

#Resultat d'execució
['pere Fantastic', 'wow pere', ['holidays with pere'], ['holidays with maria and pere'],
['holidays with maria and pere and joan', 'pere always smiling']]
```

Exercici 4 [2 punts]. BST.

La universitat de Júpiter assigna un identificador per a cada estudiant, consistent en un enter aleatori únic. Cada identificador s'emmagatzema en un arbre binari de cerca (BST), segons la definició que segueix.

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

Quan s'admet un estudiant, la universitat tria un enter al·leatori, entre el valor 0 i el valor 9000, i comprova si ja es troba en el BST. Si no hi és, s'assigna l'identificador a l'estudiant. Si hi és, es repeteix el procés amb un nou número al·leatori.

Malauradament, el sindicat de hackers frikis ha modificat el BST i n'ha canviat un o més valors. Tement el pitjor, la universitat t'ha llogat com a estudiant TIC per tal de comprovar el BST. Com a primer pas, decideixes verificar que el BST està correctament estructurat. A tal efecte cal que dissenyeu i implementeu únicament el mètode **recursiu** *desubicat*, tal que ha de comprovar que el BST emmagatzema valors correctes i que compleixen la propietat dels BST. En cas que no ho sigui, ha de retornar el primer valor erroni trobat.

Per exemple, en el BST següent hauria de retornar el valor 9999 (el 5 i el -4 també estan desubicats).