Prova final de TECPRO

18/06/2018 Grau en Enginyeria de Sistemes TIC 2H

COGNOMS:

NOM: GRUP de LAB:

Exercici 1. Donat el següent fragment de codi, i per cadascuna de les afirmacions que segueixen, escriviu si l'afirmació és [Certa/Falsa]. **(La no justificació invalida la resposta)**

```
class F(object):
    x = 0
    y = 10
    def __init__(self, y):
        self.x = y
        self.__z=99
    def a1(self, y):
        self.x = max(self.x, y)
        return self.x

    def a2(self):
        self.x = max(self.x, self.y)
        return self.x

    def b(self,a):
        return self.x+a

class G(F):
    def b(self):
        self.y = self.x * self.x
        return self.y

    def a2(self):
        self.x=0
```

```
class H(F):
    def b(self):
        self.y=22
        return self.y

class I(H):
    pass

class J(G,H):
    pass

if __name__=='__main__':
    h1=H(2)
    g1=G(2)
    v=[]
    v.append(h1)
    v.append(g1)
    for a in v:
        print a.b()
```

1. El mètode b de la classe G és un exemple de redefinició de mètodes.
2. El mètode $a2$ de la classe G és un exemple de sobrecàrrega de mètodes.
3. Un objecte de la classe G pot accedir a l'atribut x de la classe F.
4. El resultat de la execució del programa serà 4 4.
5. No es poden crear objectes de la classe I perquè no hi ha definit el mètode constructor.
6. El mètode append aplicat sobre llistes a Python té una complexitat en cas pitjor linial.
7. Un objecte de la classe H pot accedir a l'atribut z de la classe F.
8. La crida $i.b(11)$ sobre un objecte de classe I generaria un error d'execució.
9. Les crides

   ```
   j=J(1); print j.b()
   ```

   provocaran un error d'execució.
10. El millor algorisme d'ordenació d'una llista de n objectes és el mètode merge sort perquè té complexitat en cas pitjor $O(n)$.

Exercici 2. Se us demana la gestió simplificada de les multes per retorn de préstecs fora de termini d'una biblioteca. En concret, se us passa l'especificació de requeriments que ha de complir l'aplicatiu. Noteu que per simplificar, s'usen tipus bàsics en la representació de llibres, persones i dates.

- Un llibre es representa per un string corresponent al seu títol.
- Una persona que usa la biblioteca es representa per un string corresponent al seu nick.
- Una data es representa per un enter, que correspon al número de dies des que es va obrir la biblioteca.

Llegiu atentament les especificacions dels mètodes i els exemples de funcionament proporcionats a continuació per tal d'aconseguir un disseny òptim de l'aplicatiu.

La classe **Biblioteca** ha de contenir un atribut anomenat **diamulta**, inicialitzat amb el valor 0.25, i els següents mètodes, que heu d'implementar en cada apartat.

[**Apartat a**] Mètode **constructor**: Donada una llista de llibres, inicialitza la biblioteca. Se us demana que utilitzeu un **diccionari** per la gestió dels continguts de la biblioteca.

[**Apartat b**] Mètode **lloga**: enregistra el llibre, la persona i la data en la que el llibre es lloga. Cada llibre pot estar en préstec fins a 7 dies. A partir de llavors direm que està endarrerit de préstec. Per exemple, si es lloga el dia x, es troba en préstec fins el dia x+7, i es trobarà endarrerit de préstec el dia x+8. Aquest mètode no retorna res.

[**Apartat c**] Mètode **retorna**: donat un llibre i la data en que el llibre es retorna, modifica el registre del llibre. Retorna un número que representa la multa si el llibre s'ha retornat fóra del termini de préstec i 0.0 si s'ha tornat en el termini previst. La multa correspon al número de dies que s'ha passat del termini de préstec multiplicat per l'atribut de classe diamulta.

[**Apartat d**] Mètode **endarrerimentLlibres**: donada una persona i una data, retorna la llista de llibres tals que la persona els ha llogat i estan endarrerits en el préstec per la data proporcionada.

A continuació segueix un exemple de funcionament. NOTA: Suposeu que les operacions efectuades són legals. Per exemple, els llibres retornats corresponen al llibres prèviament llogats. I els llibres que es lloguen, s'han retornat abans. Suposeu també que només hi ha 1 exemplar de cada llibre. Adoneu-vos que no és necessària cap gestió per als usuaris.

```
if __name__=='__main__':
    lib = Biblioteca(['Tintin', 'Matrix', 'Python', 'Java', 'MySQL', 'Biblia'])
    lib.lloga('Tintin', 'Elena', 1)
    lib.lloga('Python', 'Elena', 1)
    lib.lloga('MySQL', 'Elena', 10)
    print lib.endarrerimentllibres('Elena', 13)
    #['Python', 'Tintin']
    print lib.retorna('Tintin', 13)
    #1.25
    print lib.retorna('Python', 18)
    #2.5
    print lib.retorna('MySQL', 18)
    #0.25
```

[**Apartat e**] A continuació definiu una nova classe de nom **BibliotecaProrroga** que es comporta com la classe Biblioteca, amb la particularitat que proporciona un període de pròrroga abans de que les multes comencin a comptar. El número de dies de pròrroga s'especifiquen quan s'instancia un objecte de la classe. Per exemple,

```
    lib = BibliotecaProrroga(2, ['Tintin', 'Matrix', 'Python', 'Java', 'MySQL', 'Biblia'])
    lib.lloga('Tintin', 'Elena', 1)
    print lib.retorna('Tintin', 13)
    #0.75
```

Se us demana la implementació completa de la classe **BibliotecaProrroga**. No està permesa la repetició del codi ja definit a la classe Biblioteca. En particular, no cal repetir el càlcul de la multa.

Exercici 3. Se us proporcionen les següents definicions (incompletes) de classes. El codi proporcionat preten tracejar les assignatures aprovades per un estudiant, per tal de comprovar si ha satisfet totes les especificacions del grau (GIR) i del departament.

```
class eMITstudent(object):
    GIR = ['mats.111', 'est.011', 'fis.01', 'fis.02', 'electro.01', 'electro.02','rest1', 'rest2', 'instlab',
'prog1', 'prog2', 'prog3', 'prog4','prog5', 'prog6', 'prog7', 'prog8']

    DEPT = []

    def __init__(self, AP = []):
        self.apr = AP

    def aprova(self, assig):
        for a in assig:
            self.apr.append(a)

    def acabaReq(self, req):
        missing = []
        for r in req:
            if not r in self.apr:
                print r +' is missing'
                missing.append(r)
        return len(missing) == 0

    def acaba(self):
        # TO DO

class Course62(eMITstudent):
    DEPT = ['tec.01', 'tec.02', 'electro.03', 'tec.041', 'tec.002', 'tec.003', 'tec.004', 'tec.005', 'tec.011',
'tec.013', 'tec.033', 'tec.115', 'tec.813', 'tec.336','tec.AUT', 'tec.AUP', 'instlab']
```

Adoneu-vos que la definició d'una subclasse de eMITstudent, tal com Course62, defineix un atribut DEPT que especifica els requeriments del departament. Un exemple d'ús es proporciona a continuació.

```
if __name__=='__main__':
    Albert = Course62(['electro.01'])
    Albert.aprova(['mats.111', 'electro.02', 'fis.01', 'prog1'])
    Albert.aprova(['electro.03', 'fis.02', 'tec.01', 'prog2'])
    Albert.aprova(['tec.02', 'tec.041', 'est.011', 'prog3'])
    Albert.aprova(['tec.002', 'tec.003', 'tec.013', 'prog4'])
    Albert.aprova(['tec.004', 'tec.005', 'tec.011', 'prog5'])
    Albert.aprova(['tec.033', 'tec.115', 'tec.336', 'prog6'])
    Albert.aprova(['tec.813', 'tec.AUT', 'tec.AUP', 'prog7', 'prog8'])
    print 'acaba? ', Albert.acaba()
    Albert.aprova(['rest1', 'rest2'])
    print 'acaba? ', Albert.acaba()
```

[**Apartat a**] Se us demana la implementació òptima del mètode **acaba**. Aquest mètode ha de comprovar si ha assolit els requeriments del GIR i els requeriments del DEPT, i retorna un booleà (**True** si ha aprovat totes les assignatures requerides pel grau i pel departament).

[**Apartat b**] La implementació prèvia proporcionada no contempla el següent requeriment: Un estudiant que hagi aprovat tec.01 i tec.02, obté automàticament el requeriment del GIR anomenant instlab. Se us demana que implementeu òptimament un nou mètode que gestioni dita funcionalitat correctament. Es penalitzarà la repetició de codi.

Exercici 4. L'empresa que gestiona el navegador ChroItic està buscant programadors que els ajudin a implementar una nova característica en la seva nova versió de navegador, que permeti tracejar els tags de pàgines .html potencials per comprovar que no hi haurà errades en la visualització de les pàgines.

Una pàgina .html està formada per un seguit de tags d'obertura i tancament, com els que segueixen. El principi i final d'un document es marquen amb els tags $\langle html \rangle$ i $\langle /html \rangle$. Fixeu-vos que els tags d'obertura tenen el format $\langle nomtag \rangle$ i els de tancament tenen el format $\langle /nomtag \rangle$. Adoneu-vos que es poden obrir diversos tags a la vegada, però que els tags de tancament han de seguir un ordre.

```
<html> <title>Prova </title> <b> <i>TecproExam </i> </b> </html>
```

En aquest cas, es crea una pàgina hmtl amb un títol i el contingut TecproExam en cursiva i negreta. En el següent exemple s'ha omès el títol de la pàgina, però la seqüència d'obertura i tancament de tags és correcta.

```
<html> <b> <i>TecproExam </i> </b> </html>
```

Exemples de pàgines incorrectes, serien per exemple,

```
<title>Prova <html> </title> <b> <i>TecproExam </i> </b> </html>
#Errònia: el tancament del tag title està en ordre incorrecte

<html>Prova <b>TecproExam </b> #Errònia: no s'ha tancat el tag html
```

Per simplificar, aplicarem els següents supòsits:

1. El separador entre tags serà l'espai en blanc
2. Les paraules entre els tags no contenen espais en blanc
3. Els tags estan ben construïts. És a dir no tindrem tags amb el format $<< nomTag >$ per exemple.

[Apartat a] Dissenyeu òptimament la funció booleana **tagsOK**, tal que donada una seqüència de strings, retorni True si el seu conjunt de tags conforma correctament una pàgina web.

NOTA: Podeu utilitzar directament la classe Stack, amb els seus mètodes associats.

```
def tagsOK(s):
    """
    >>> tagsOK('<html> <title>Prova </title> <b> <i>TecproExam </i> </b> </html>')
    True
    >>> tagsOK('<html> <title>Prova </title> <b> <i>TecproExam </b> </i> </html>')
    False
    >>> tagsOK('<html> <title>Prova <b> <i>TecproExam </b> </i> </html>')
    False
    >>> tagsOK('<html> <title>Prova </title> <b> <i>TecproExam </i> </b> ')
    False
    """
```

[Apartat b] Quina és la complexitat en cas pitjor de la funció **tagsOK**? Utilitza la notació O, i justifica breument el seu cost.

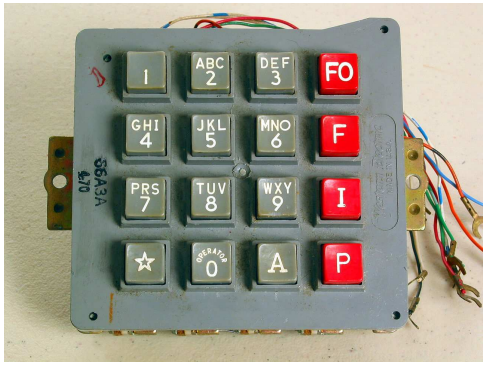

Figure 1: Touch-ToneTM(Font: Wikipedia)

Exercici 5. En un teclat de telèfon que segueixi l'estàndard Touch-ToneTM, els dígits es mapegen en l'alfabet (excepte les lletres Q i Z) tal i com mostra la Figura 1.

Per tal de fer els números de telèfons més fàcils de memoritzar, als proveidors de servei telefònic els agrada trobar paraules (mnemònics) que facin que el número sigui fàcil de recordar. Per exemple, el número 6378687 pot ser recordat fàcilment per un dels seus mnemònics, NERVOUS.

Suposeu que heu estat llogats per una companyia telefònica local amb l'objectiu de dissenyar i implementar òptimament la funció **recursiva llistarMnemonics**, que generarà totes les possibles combinacions de lletres corresponents a un número donat, representat com a un string de dígits. Per exemple, la crida a **llistarMnemonics("723")** ha de generar les següents 27 combinacions possibles de lletres corresponents al prefix 723.

```
PAD PBD PCD RAD RBD RCD SAD SBD SCD
PAE PBE PCE RAE RBE RCE SAE SBE SCE
PAF PBF PCF RAF RBF RCF SAF SBF SCF
```

NOTA: No està permés l'ús d'slices/llesques ni funcions predefinides de Python. Les úniques operacions permeses són len, concatenació i accés a un element d'un string. La resolució del problema mitjançant una funció no recursiva tindrà una puntuació nul·la.

Exercici 6. BST. [**Apartat a**] Donat un arbre binari de cerca que emmagatzema valors enters, i dos dos valors enters k1 i k2 (on $k1 < k2$), se us demana que implementeu òptimament el mètode recursiu **mostraRang**, tal que ha de mostrar per pantalla les claus del l'arbre entre el rang k1 i k2, és a dir, mostra tots els valors de l'arbre entre k1 i k2 (ambdòs valors inclosos). Cal mostrar la informació en ordre creixent. Per exemple, donat l'arbre de la Figura 2, si k1=10 i k2=22, el mètode ha d'escriure 12, 20, 22.

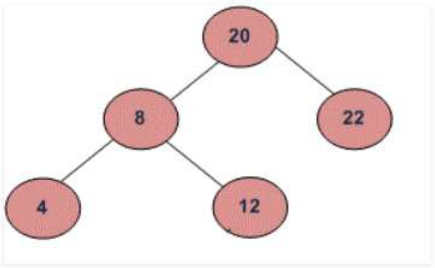

Figure 2: A Binary Search Tree

Suposeu la següent definició d'arbre binari de cerca i els exemples de funcionament que segueixen.

```
class BST(object):
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None
    def mostraRang(root, k1, k2):
        #TO DO
```

```
if __name__=='__main__':
  k1 = 10 ; k2 = 25 ;
  arbre = BST(20)
  arbre.left = BST(8)
  arbre.right = BST(22)
  arbre.left.left = BST(4)
  arbre.left.right = BST(12)
  mostraRang(arbre, k1, k2)
```

[**Apartat b**] Quina és la complexitat en cas pitjor del mètode **mostraRang** implementat? Utilitza la notació O, i justifica breument el seu cost.