PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 – Programación Avanzada

Examen

Martes 4 de Julio 2017

- Duración: 2,5 horas.
- El examen tiene 120 puntos.
- Podrá hacer preguntas cada 30 minutos en voz alta. Solo se responderá preguntas que ayuden a mejorar la comprensión del enunciado.
- Conteste cada pregunta en hojas separadas, es decir, a lo más una pregunta por hoja. Las hojas de respuesta están incluidas con el enunciado. Puede solicitar hojas adicionales a los ayudantes en caso que sea necesario.
- Si no responde una pregunta, entregue una hoja en blanco con su nombre, número de alumno, sección y número de pregunta.
- Una declaración de clases y métodos corresponde a

```python
class A:
    def __init__(self, param1):
        self.param1 = param1

    def metodo_1(self, param2):
        # param2: corresponde a Y
        # return: el resultado de hacer multiplicar self.param1 y param2
        # Entrega la multiplicacion entre self.param1 y param2
        pass
```

- La descripción del flujo de un método es una explicación detallada de lo que hace el método. Por ejemplo, para el siguiente método:

```python
def multiplicar(a, b, c):
    aux = a * b
    aux = aux * c
    return aux
```

Una buena descripción sería: *"Se multiplican a y b, se guardan en una variable, luego esta se multiplica por c y se retorna ese resultado"*. Una mala descripción sería: *"Se multiplican a, b y c"*.

1. **(45 pts)** El refugio de animales XX recibe perros y gatos abandonados, los cuidan y les buscan hogares temporales hasta que encuentren hogar definitivo. Esta fundación necesita saber cuántas horas de voluntarios de cada tipo necesita en cada horario, calcular la cantidad de comida para gatos y perros, y calcular cuánto dinero para veterinario y sicólogo necesitará por mes.

   A la única sede que tienen llega en promedio un animal cada 24 horas. Todos los animales que llegan se aceptan. La probabilidad de que sea perro es de 70 % y de que sea gato un 30 %. La edad del animal distribuye normal con media 23 meses y desviación estándar 5 meses. El tamaño también distribuye normal con media 10 kgs y desviación estándar 4 kg. Cuando el animal llega, debe quedar aislado del resto de los animales hasta que sea revisado por el veterinario. Durante ese período solo puede dejar su jaula para hacer sus necesidades 3 veces al día y para jugar tres horas con un voluntario.

   El veterinario va como mínimo una vez a la semana y cada vez que hay 10 animales nuevos. Él revisa a todos los animales nuevos y a los antiguos que pueden estar enfermos. Existe un 30 % de probabilidad de que un animal revisado por el veterinario esté enfermo y un 20 % de probabilidad que un animal nuevo sea no sociable. Si un animal está enfermo, el veterinario ordena su traslado a la clínica veterinaria hasta que este se recupere. La cantidad de días la determina el veterinario y puede ser entre 3 y 10. Cada día en la clínica veterinaria cuesta $25.000, lo que incluye remedios, comida y cuidados. Si el animal está sano, comienza a realizar la rutina de los otros animales del refugio. Por cada animal que revisa el veterinario se pagan $10.000.

   Hay dos tipos de voluntarios: novatos y expertos. Los expertos son los únicos que evalúan diariamente a los animales y entrenan a los animales no sociables. Durante la noche debe haber un voluntario experto por cada 30 animales y un no-experto por cada 15 animales. Todos los voluntarios son padrinos de uno a cinco animales. Un animal puede tener más de un padrino y como mínimo debe tener uno.

   Cada animal come entre 100 y 200 gramos de alimento al día. Esta cantidad se determina cuando el animal llega al refugio y depende de la edad y el tamaño. Un voluntario se demora 2 minutos en servirle comida a un animal.

   Los animales tienen una rutina:

   (1) entre las 7 y las 8 se les sirve el desayuno.

   (2) entre las 9 y las 11 los voluntarios realizan las evaluaciones individuales. Si el animal no pasa la evaluación se debe agregar a la lista para que sea evaluado por un veterinario y debe seguir el mismo protocolo que el de un animal que acaba de llegar. Cada evaluación demora 5 minutos.

   (3) entre las 11 y las 19 hrs los animales más sociables se dejan salir al sector común. Aquí debe haber un voluntario cada 5 animales supervisando que los animales no tengan problemas. Existe una probabilidad de un 2 % que dos animales al azar peleen durante este tiempo. Esos animales se encierran y deben ser entrenados una hora por voluntarios expertos el día siguiente para poder volver al área común. Los menos sociables se deben entrenar durante este horario. Los voluntarios más expertos deben entrenar individualmente por 3 hrs a estos animales.

   (4) Se sirve el almuerzo entre las 13 y 14 hrs en el sector común para los animales sociables y en sus jaulas para los animales no sociables.

   (5) Entre las 19 y 20 hrs se sirve la última comida del día en la jaula de cada animal.

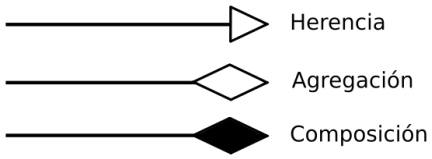
   (6) Desde las 20hrs, el veterinario o el sicólogo revisa a los animales que lo necesiten.

   Los animales que no son sociables se evalúan una vez al mes para ver sus avances en sociabilización. Esta evaluación la realiza un sicólogo de animales. Cada evaluación cuesta $30.000. Un animal tiene

una probabilidad de ser evaluado como sociable de 40 %. Por cada 10 horas de entrenamiento esta probabilidad aumenta en 1.

La fundación le ha pedido a usted realizar una simulación para poder organizarse mejor, según los objetivos descritos en el primer párrafo.

(a) **(15 pts)** Construya el diagrama de clases `UML` que modela este problema. Indique claramente las clases y las relaciones entre estas (composición, agregación y herencia) y los métodos y atributos que estime conveniente. En caso que no sea claro cuál de las relaciones existe entre dos clases, justifique su respuesta (en una línea es suficiente). Realice los supuestos que crea necesarios. Recuerde que los símbolos en UML son:



(b) **(10 pts)** Escriba la declaración de clases y métodos según el diagrama realizado en (a). Los atributos de cada clase deben estar declarados en el método `__init__` de la clase. No olvide ser coherente con el diagrama de clases que realizó en (a).

(c) **(5 pts)** ¿Cuáles son las estadísticas que necesitará calcular?

(d) **(15 pts)** Seleccione 5 eventos, al menos uno relacionado con cada entidad, y escriba para cada uno: nombre descriptivo, cuándo ocurre o se gatilla, consecuencias que tiene y cómo afecta a las estadísticas que describió en la pregunta (c).

---

2. **(30 pts)**. Considere la siguiente estructura de datos para un grafo **dirigido** que representa a los miembros de una red social (nodos) y la relación *sigue a* (aristas dirigidas) entre miembros de la red.

   Cada miembro m posee una dirección IP donde se encuentra ejecutando, y posee un muro donde se muestran los mensajes de todos los contactos de m. Para comunicarse con otro miembro debe mandarse un mensaje a su dirección IP.

```python
class Miembro:

    def __init__(self, nombre, host):
        self.nombre = nombre
        self.ip = ip
        self.contactos = []
        self.muro = []
        self.estado = ""

    def agregar_contacto(self, miembro):
        self.contactos.append(miembro)

    def __repr__(self):
        return self.nombre + " -> (" + ",".join(
            [c.nombre for c in self.contactos]) + ")"
```

(a) **(10 pts)** Escriba una función `recomendar(m)` que reciba como parámetro un objeto `m` de la clase `Miembro`, y retorna una lista de contactos para recomendar a ese miembro. Los contactos

a recomendar para un miembro m son aquellos miembros que son seguidos por los contactos de m, pero a los cuales m no sigue. **Solo para esta parte** asuma que toda la información de los contactos de cada miembro se encuentra disponible en la estructura de datos; esto significa que no necesita envía mensajes por la red.

(b) **(10 pts)** Explique cómo conectaría los miembros de esta red social. Describa todos los métodos que le faltan a `Miembro` para poder conectarse con la red.

(c) **(10 pts)** Describa el funcionamiento de un *thread* que permite publicar mensajes de estado. Cuando m publica un mensaje de estado, éste debe aparecer en los muros de aquellos miembros que tienen a m como contacto. Describa qué elementos incluiría en un mensaje y como manejaría el hecho que el muro puede recibir múltiples mensajes simultáneamente de distintos miembros.

---

3. (a) (**15 pts**) Indique qué imprime al finalizar el siguiente código:

```
1  import re
2
3  def aplicar_a(fn):
4      def _aplicar_a(msg):
5          l = re.split('(<[^>]+>)', fn(msg))
6          l = [
7               '<b>{}</b>'.format(e) \
8                      if re.match('[0-9]', e) else e for e in l
9              ]
10         return ''.join(l)
11     return _aplicar_a
12
13
14 def aplicar_b(arg):
15     def _aplicar_b(fn):
16         def __aplicar_b(msg):
17             b = ["<{0}>{1}</{0}>".format(arg, fn(a)) for a in msg.split()]
18             return "".join(b)
19         return __aplicar_b
20     return _aplicar_b
21
22 @aplicar_a
23 @aplicar_b("u")
24 def mensaje(msg):
25     return msg
26
27 print(mensaje("Este mensaje incluye 4 espacios"))
```

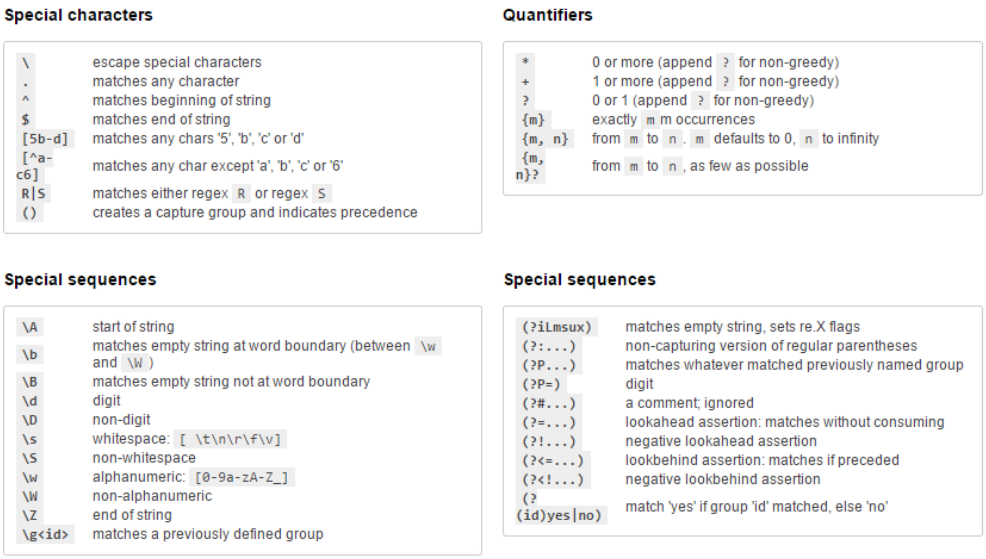


Figura 1: Resumen de caracteres usados en regex.

(b) (**5 pts**) En el siguiente código, ¿qué clase corresponde a un iterable y cuál corresponde a un iterador? Justifique.

```
class ClassX:
    def __init__(self, objet):
        self.objet = objet

    def __iter__(self):
        return ClassY(self.objet)

class ClassY:
    def __init__(self, data):
        self.data = data

    def __iter__(self):
        return self

    def __next__(self):
        if self.data is None:
            raise StopIteration("The End")
        else:
            current = self.data
            self.data = self.data.f
            return current
```

(c) (**5 pts**) De acuerdo al siguiente código, ¿qué ocurrirá al ejecutar el init de un clase que tiene como metaclase a `MetaObject`? Explique línea por línea.

```
import random
```

```
2
3  class MetaObject(type):
4
5      def __new__(meta, name, bases, attrs):
6          o_ = attrs['__init__']
7
8          def n_(self, *args, **kwargs):
9              self.id_ = random.randint(25, 65)
10             o_(self,args, kwargs)
11
12         attrs['__init__'] = n_
13         attrs['instances'] = {}
14
15         return super().__new__(meta, name, bases, attrs)
16
17     def __call__(cls, *args, **kwargs):
18         if args in cls.instances:
19             return None
20         else:
21             obj = super().__call__(*args, **kwargs)
22             cls.instances[args] = obj
23             return obj
```

(d) (**5 pts**) De acuerdo al código de la pregunta anterior, ¿qué ocurrirá al crear un objeto de una clase que tiene como metaclase a `MetaObject`?

(e) **(5 pts)** Indique qué imprime el siguiente código.

```
1  import threading
2  import time
3
4
5  def f1(lock1, lock2):
6      with lock1:
7           print('Primera parte - f1')
8           time.sleep(10)
9      with lock2:
10          print('Segunda parte - f1')
11          time.sleep(5)
12
13 def f2(lock1, lock2):
14     with lock2:
15          print('Primera parte - f2')
16          time.sleep(10)
17     with lock1:
18          print('Segunda parte - f2')
19          time.sleep(5)
20
21 a_lock = threading.Lock()
22 b_lock = threading.Lock()
23
```

```
t1 = threading.Thread(target=f1, args=(a_lock, b_lock))
t2 = threading.Thread(target=f2, args=(b_lock, a_lock))
t2.start()
t1.start()
```

---

4. **(10 pts)** En no más de diez líneas, responda solo **CINCO** de las siguientes preguntas (2 puntos cada pregunta). En caso de que conteste más preguntas, **SOLO** serán corregidas las primeras 5 respuestas que se encuentren en su hoja de respuesta. Sea breve y conciso.

   (a) ¿Que uso tiene el argumento `object_hook` durante la deserialización de un objeto JSON?

   (b) ¿Cuál es la salida del siguiente código?

```
a = ['K1', 'K2', 'K3']
b = ['124', '45', '4']
l = {k: v for k, v in zip(a, b)}
m = dict(map(lambda k: (k[0], len(k[1])), l.items()))
n = reduce(lambda x, y: x+y, list(filter(lambda x: x > 2, m.values())))
print(n)
```

   (c) ¿Para qué son utilizados los *context managers* en Python y de qué forma es posible personalizarlos en una clase cualquiera?

   (d) En `unittest`, ¿cuál es la diferencia entre `setUp` y `tearDown`?

   (e) Explique la diferencia entre `re.search` y `re.match`.

   (f) Explique por qué no es bueno capturar excepciones sin explicitar el tipo de excepción.

   (g) ¿Cuáles son las ventajas y desventajas de serializar con `pickle`?

   (h) ¿Por qué es útil hacer la diferencia entre *backend* y *frontend*?

   (i) ¿Qué es ducktyping?

Nombre: **N° de alumno:** **Sección:**

Nombre: **N° de alumno:** **Sección:**

Nombre: **N° de alumno:** **Sección:**

Nombre: **N° de alumno:** **Sección:**

Nombre: **N° de alumno:** **Sección:**

Nombre: **N° de alumno:** **Sección:**