Seminario de Python - 2012
Parcial 3ra fecha 6/8

Parte Práctica (se aprueba con 50 pts)

1.- (30 pts.) Se desea almacenar información referida a las medallas obtenidas por cada país en los Juegos Olímpicos. Para ello se tendrá un archivo con datos de todos los países que tienen alguna medalla, discriminando por cada deporte, tipo de medalla obtenida (oro, plata, bronce) y cantidad. Realizar el alta de una medalla para un deporte y país determinado y luego obtener un listado de la cantidad de medallas totales obtenidas por un país en cuestión agrupadas por su tipo.

Nota 1: Utilice manejo de excepciones.
Nota 2: Indique claramente las estructuras de datos utilizadas, dando un ejemplo con datos en las estructuras.
Nota 3: Puede utilizar pickle.

2.- (30 pts.) Se desea implementar con POO un Blog (el cual tiene un nombre y descripción) en donde se tienen usuarios que pueden realizar publicaciones. Cada publicación pertenece a una categoría. Existe un administrador del blog que puede crear o eliminar categorías (también puede crear publicaciones al igual que el usuario). En cuanto a la eliminación se debe comprobar que no exista ninguna publicación que pertenezca a dicha categoría. Los usuarios y el administrador poseen un nombre, apellido, nombre de usuario y obviamente sus publicaciones.
Se debe modelar el diagrama de clases UML y la funcionalidad mínima descripta anteriormente.
Por último se deben crear al menos 2 categorías, 1 publicación por parte de un usuario y otra por parte del administrador. Luego el administrador debe intentar eliminar una categoría que tenga asociada una publicación y otra que no.

3.- (20 pts.) Explique detalladamente qué realiza el siguiente juego, cómo se crea, formas de interacción, terminación, sugerencias para mejorarlo, etc:

```
import sys
import pygame
from pygame.locals import *

class MiSprite(pygame.sprite.Sprite):

    def __init__(self, dibujo, posX, posY, velocidad):
        pygame.sprite.Sprite.__init__(self)
            self.image = pygame.image.load(dibujo)
            self.image = self.image.convert()
            self.rect = self.image.get_rect()
            self.rect.topleft = (posX, posY)
            self.vel = velocidad

            def update(self):
                self.rect.move_ip(self.vel,0)
                if self.rect.left < 0 or self.rect.right > 640:
```

```
            self.vel = -self.vel
            self.image = pygame.transform.flip(self.image, True, False)

def update_clock():
    ticks = pygame.time.get_ticks()
    font = pygame.font.SysFont("arial", 36)
    text = font.render(str(ticks), True, (0, 128, 10))
    textpos = text.get_rect(centerx=pantalla.get_width()/2)
    pantalla.blit(text, textpos)

pygame.init()
pantalla = pygame.display.set_mode((640, 480))
personaje1 = MiSprite("mario.bmp", 0, 350, 5)
grupo = pygame.sprite.RenderUpdates(personaje1)
personaje2 = MiSprite("mario.bmp", 0, 250, 6)
grupo.add(personaje2)
reloj = pygame.time.Clock()

while True:
    reloj.tick(60)
    for evento in pygame.event.get():
        if evento.type == QUIT or (evento.type == pygame.KEYDOWN and evento.key ==
pygame.K_ESCAPE):
            pygame.quit()
            sys.exit()
        grupo.update()
        pantalla.fill((255,255,255))
        grupo.draw(pantalla)
        update_clock()
        pygame.display.update()
```

Ayuda:

- La función *flip()* del módulo **transform** permite rotar una imagen vertical y horizontalmente.
- La función *get_ticks()* del módulo **time** devuelve la cantidad de milisegundos que pasaron a partir de iniciado el juego.