

Programación—Certamen 2, sábado 19 de noviembre de 2011

Nombre:

Rol: -

1. [25%] Indique qué es lo que imprimen los siguientes programas.

```
d = {'a': 'b', 'b': 'c'}  
print d['b']
```

```
k = ['a', 'b', 'b', 'c']  
print k[1]
```

```
p = (3, 1, 2)  
q = (3, 1)  
print p < q
```

```
x, y = ((2, 8), 16)  
z, w = x  
print y + w
```

```
a = 'estanque'  
b = 'tan'  
c = b in a  
d = a[3] in b  
print c and d
```

```
print len({  
    0: ['a', 'bcd'],  
    1: ['ef', 'ghij', 'k'],  
    2: ['l'],  
}[1][1][1])
```

```
def f(x, y, z):  
    return x + y * z
```

```
w, i = (0, 1)  
a = f('w', 'i', 2)  
b = f(w, i, 2)  
print a * b
```

```
def f(d):  
    return d[1]
```

```
a = f('pala')  
b = f({2: 3, 1: 4})  
print a * b
```

Programación—Certamen 2, sábado 19 de noviembre de 2011

Nombre:

Rol:

2. [25%] Escriba las funciones necesarias para que el siguiente programa funcione.

```
libros = [  
    ('Papelucho programador', 'Marcela Paz', 1983),  
    ('Don Python de la Mancha', 'Miguel de Cervantes', 1615),  
    ('Raw_input y Julieta', 'William Shakespeare', 1597),  
    ('La tuplamorfosis', 'Franz Kafka', 1915),  
    # ...  
]  
  
datos_autores = {  
    # autor: nacimiento, defuncion, idioma  
    'William Shakespeare': ((1564, 4, 26), (1616, 5, 3), 'inglés'),  
    'Franz Kafka': ((1883, 7, 3), (1924, 6, 3), 'alemán'),  
    'Marcela Paz': ((1902, 2, 28), (1985, 6, 12), 'español'),  
    'Miguel de Cervantes': ((1547, 9, 29), (1616, 4, 22), 'español'),  
    # ...  
}  
  
titulo = raw_input('Ingrese titulo del libro: ')  
print 'El libro fue escrito en', obtener_idioma(titulo),  
print 'por', obtener_autor(titulo)  
print 'El autor fallecio', calcular_anos_antes_de_morir(titulo), 'años',  
print 'después de haber escrito el libro'
```

Programación—Certamen 2, sábado 19 de noviembre de 2011

Nombre:

Rol: -

3. [25 %] La ciudad de Pitonia tiene una alta congestión de vehículos circulando por sus calles. Las autoridades han decidido aplicar restricción vehicular para descongestionar la ciudad en base a las patentes de los vehículos.

La patente de un vehículo es un string de cuatro letras y dos dígitos, y la restricción depende sólo del *penúltimo* dígito. Por ejemplo, para la patente GEEA78, el dígito 7 es el utilizado para evaluar la restricción.

La restricción vehicular de los días hábiles de la semana se encuentra ingresada en el diccionario `digitos`, cuyas llaves son los días de la semanas, y cuyos valores son tuplas de cuatro enteros que representan los dígitos con restricción de ese día.

- Implemente la función `esta_con_restriccion(digitos, dia, patente)`, que indique si el vehículo está o no con restricción.
- Implemente la función `dias_con_restriccion(digitos, patente)`, que retorne la lista de los días en que el vehículo no puede circular.
- Implemente la función `dias_sin_restriccion(digitos, patente)`, que retorne el conjunto de los días en que el vehículo sí puede circular.

```
>>> digitos = {'lunes': (3, 4, 5, 6), 'martes': (7, 8, 9, 0),
...           'miercoles': (1, 2, 3, 4), 'jueves': (5, 6, 7, 8),
...           'viernes': (9, 0, 1, 2)}
>>> esta_con_restriccion(digitos, 'lunes', 'BBDT35')
True
>>> dias_con_restriccion(digitos, 'BBDT35')
['lunes', 'miercoles']
>>> dias_sin_restriccion(digitos, 'BBDT35')
set(['jueves', 'martes', 'viernes'])
```

Programación—Certamen 2, sábado 19 de noviembre de 2011

Nombre: Rol: -

4. [25%] La empresa de telecomunicaciones Python Está Aquí desea implementar un programa para monitorear y controlar la asignación de antenas para los teléfonos móviles de sus clientes en una zona específica de la ciudad. Para ello se considera la siguiente representación:

- la posición de cada antena es una tupla $(a, (x, y))$, donde a es el nombre de la antena y (x, y) es su ubicación en la zona, dada en kilómetros;
- la posición de casa cliente es una tupla $(c, (x, y))$, donde c es un identificador del cliente y (x, y) su ubicación en la ciudad, dada en kilómetros.

El máximo radio de cobertura de una antena es de 3 kilómetros.

Implemente la función `mejor_antena(antenas, clientes, c)`, cuyos parámetros son las listas de antenas y clientes, junto con el identificador c de un cliente en particular. La función debe retornar el nombre de la antena que entrega la mejor cobertura (la más cercana) al cliente. Si existe más de una, elija cualquiera. Si no hay antenas dentro del radio de cobertura, debe retornar **None**.

```
>>> antenas = [('A1', (1, 7)), ('A2', (5, 2)), ('A3', (4, 8))]
>>> clientes = [('C1', (1, 2)), ('C2', (4, 1)), ('C3', (3, 5)),
...            ('C4', (3, 9)), ('C5', (5, 7))]
>>> mejor_antena(antenas, clientes, 'C4')
'A3'
>>> mejor_antena(antenas, clientes, 'C1')
None
```