Programación—Certamen Recuperativo (CC) - Miércoles 23 de Septiembre de 2015

Nombre: _____ Rol: _____ – _____

1. **[25%]** Asigne valores a las variables `a` y `b` para que los siguientes códigos impriman el output indicado.

a)

```
a =
b =
if not a:
    print a
if b != a:
    print b
else:
    print not a
```

Output:

```
False
True
```

b)

```
a =
b =
while b < a:
    if a - 2 > b:
        print a,
    else:
        print b
    b = b + 1
```

Output:

```
6 6 6 6 4
5
```

c)

```
def f1(a, b):
    if a % b == 0:
        return 0
    return b % a

a =
b =
print f1(b, a)
```

Output:

```
7
```

d)

```
def f2(m, n):
    c = '#progralover'
    i = 0
    c2 = ''
    while i < m:
        if c[i] not in n:
            c2 = c2 + c[i]
        else:
            c2 = c2 + n[-2:]
        i = i + 1
    return c2

a =
b =
print f2(a, b)
```

Output:

```
#prograisover
```

Programación—Certamen Recuperativo (CC) - Miércoles 23 de Septiembre de 2015

Nombre: Rol: –

2. **[35 %]** Luego de la primera entrega del reloj inteligente PyWatch no se tuvo el éxito de ventas esperado, por lo tanto se quiere agregar nuevas funcionalidades a la siguiente versión. El servidor de PyApple almacena información de los usuarios de PyWatch tomada desde cada aparato. Esta información corresponde a la presión arterial de cada usuario en mediciones sucesivas. En el servidor de PyApple es posible acceder a la información de los usuarios de pyWatch hasta cierto instante a través del diccionario personas:

```
personas = {
'H2010': [(119, 79), (141, 93), (140, 90), (133, 81)],
'H2023': [(140, 93), (140, 90), (119, 78), (140, 90)],
# ...
}
```

Este diccionario contiene información de muchos usuarios, en particular existen dos cuyos códigos son `'H2010'` y `'H2023'`. Para el usuario `'H2010'`, los valores asociados al código son las mediciones sucesivas de presión. La primera medición es `(119, 79)`, donde 119 es la presión arterial sistólica y 79 la diastólica.

Con esto, los ingenieros de PyApple le piden a usted implementar las siguientes funciones:

a) La función `hipertensos(personas)` que recibe la estructura `personas` y retorna un diccionario con llave el usuario y valor la cantidad de alzas de los usuarios hipertensos. Una persona es hipertensa cuando ha experimentado al menos un alza de presión, es decir, cuando ambos valores de presión sistólica y diastólica son mayores o iguales que `(140, 90)` respectivamente.

```
>>> hipertensos(personas)
{'H2010': 2, 'H2023': 3}
```

b) La función `maximo_usuarios(personas, indice)` que recibe el diccionario `personas` y un índice (0 o 1). La función retorna un conjunto de tuplas con el código del usuario y su máxima presión según el `indice` (0: sistólica o 1: diastólica).

```
>>> maximo_usuarios(personas, 0)
set([('H2010', 141), ('H2023', 140)])
>>> maximo_usuarios(personas, 1)
set([('H2010', 93), ('H2023', 93)])
```

c) La función `orden_hipertensos(personas)` que recibe el diccionario `personas` y retorna una lista con los ids de los usuarios hipertensos ordenados de menor a mayor número de alzas.

```
>>> orden_hipertensos(personas)
['H2010', 'H2023']
```

Programación—Certamen Recuperativo (CC) - Miércoles 23 de Septiembre de 2015

Nombre: Rol: –

3. **[40 %]** Complementando la pregunta 2, se le solicita:

a) La función `nuevo_registro(usuario, personas, fecha)` que agregue una nueva línea al archivo `[usuario].dat` con la `fecha` indicada y los valores de las presiones de `usuario` almacenadas en `personas`. Note que el nombre del archivo está dado por el código del usuario. Cada línea del archivo tiene el siguiente formato: `fecha:presiones`, donde `presiones` son las presiones separadas por `@` (`sistolica,diastolica`).

```
>>> nuevo_registro('H2023',
   personas, '2015/09/10')
>>>
```

H2023.dat

```
2015/09/09:137,93@140,85@123,75
2015/09/10:140,93@140,90@119,78@140,90
```

En el ejemplo, el archivo `H2023.dat` tenía un registro previo con fecha `2015/09/09`.

b) La función `obtener_registro(usuario, fecha)` que retorne una lista con las mediciones de presión (como **números enteros**) del `usuario` en la `fecha` indicada. Si la fecha no existe, retornar una lista vacía.

```
>>> obtener_registro('H2023', '2015/09/10')
[(140, 93), (140, 90), (119, 78), (140, 90)]
```

c) La función `contar_registros(personas)` que retorne un diccionario con llave el usuario y valor la cantidad de fechas registradas en su archivo de registros. Asuma que existe un archivo por usuario.

```
>>> contar_registros(personas)
{'H2023': 2, 'H2010': 0}
```