Programación—C.Recuperativo - Jueves 30 de Junio de 2016

Nombre

Rol — Paralelo

1. [20 %] Realice el ruteo del siguiente programa e indique qué es lo que imprime. Cada vez que el valor de una variable cambie, escríbalo en una nueva fila de la tabla. Recuerde que si una variable es de tipo string, su valor debe ir entre comillas simples ' '.
***Importante**: La tabla tiene suficientes filas.*

```
def funcion(x,y,z):
    l = []
    if 0 < abs(x) < len(z):
        if x < 0:
            x = len(z) + x
        for i in range(len(z)):
            if x == i:
                l.append(y)
            l.append(z[i])
        return l
    return False

a = [1, 4, 8]
a = funcion(-2, 3, a)
print a
```

global	funcion				
a	x	y	z	l	i

Análisis de Algoritmo

En base a la `funcion` descrita arriba, analice y determine, **en pocas palabras**, lo que realiza. No debe usar más del espacio indicado.

Programación—C.Recuperativo - Jueves 30 de Junio de 2016

Nombre

Rol — Paralelo

2. **[40 %]** El grupo de música, **Iron Pyhton**, ha decidido iniciar una serie de conciertos en diferentes lugares del país, los cuales están almacenados en un conjunto de tuplas compuestas por un *string* que representan el nombre de la ciudad y un número entero con la capacidad de público de cada locación. La siguiente estructura describe lo mencionado:

```
conciertos = set([('Pythonia', 2000), ('Santiago', 15000), ('
   Pelotillehue', 4000), ('Valparaiso', 5000), ('Vina del mar', 3000),
   #... ])
```

Además, la banda posee un `diccionario` que guarda información sobre las entradas de los respectivos conciertos. Para ello, la estructura mencionada posee como llave el nombre de la ciudad y su valor es una tupla con la cantidad de entradas vendidas y el valor de cada una de ellas.

```
entradas = {'Pythonia': (2000, 6000), 'Santiago': (7000, 8000),
            'Pelotillehue': (100, 4000), 'Valparaiso': (1000, 4000),
            'Vina del mar': (200, 5000), # ...
            }
```

a) Desarrolle la función `menor_publico(conciertos)` que recibe como parámetro el conjunto de tuplas `conciertos`. La función debe retornar una tupla, compuesta por el nombre de la ciudad y su capacidad de público, de la ciudad con menor capacidad de público.

```
>>> menor_publico(conciertos)
('Pythonia', 2000)
```

b) Desarrolle la función `gira(conciertos)` que recibe como parámetro el conjunto de tuplas `conciertos`. La función debe retornar una lista con el nombre de las ciudades ordenadas de menor a mayor capacidad de público.

```
>>> gira(conciertos)
['Pythonia', 'Vina del mar', 'Pelotillehue', 'Valparaiso', 'Santiago']
```

c) Desarrolle la función `nueva_gira(conciertos, entradas, minimo)` que recibe como parámetro el conjunto de tuplas `conciertos`, el diccionario `entradas` y un valor entero `minimo`. La función debe calcular qué conciertos son viables monetariamente respecto al monto mínimo entregado como parámetro de acuerdo a la recaudación (*entradas* $*$ *precio*) que han obtenido. Finalmente, la función debe retornar una lista con los nombres de las ciudades ordenadas ascendentemente respecto a su capacidad de público.

```
>>> nueva_gira(conciertos, entradas, 1000000)
['Pythonia', 'Valparaiso', 'Santiago']
```

Programación—C.Recuperativo - Jueves 30 de Junio de 2016

Nombre

Rol — Paralelo

3. **[40 %]** En una galaxia muy muy lejana, el lado oscuro de la fuerza vuelve a atacar a través de *la nueva orden*.

Usted es Poe Dameron, el mejor piloto de la rebelión, y ha sido enviado por la General Organa a bombardear un puesto de avanzada de *la nueva orden*. El droide *BB 8* le ha provisto de un archivo de texto con las posiciones de los enemigos en el siguiente formato: `x,y;nombre_unidad;clase_unidad`.

El archivo `primera_orden.csv` es un ejemplo de lo anterior.

`primera_orden.csv`

```
3,1;stormtrooper;GU
4,4;flametrooper;GU
5,0;riot control;GU
2,2;sith lord;FU
1,1;tie fighter;SS
```

Desarrolle la función `disparar(ubicacion, radio, nombre_archivo, clases)` que recibe como parámetro la tupla `ubicacion` con el lugar donde se lanzará el un misil, el número real `radio` que representa el radio de alcance de una explosión circular, un string `nombre_archivo` con el nombre del archivo con la información del enemigo y la lista de string `clases` que posee las clases por las que se está consultando. La función debe **actualizar** el archivo recibido como parámetro, eliminando **todos** los enemigos que han sido destruidos por el impacto de la bomba y además debe retornar un diccionario cuyas llaves sean las clases por las que se están consultando y el valor la cantidad de enemigos de esas clases que fueron destruidos por el impacto de la bomba.

Nota: Una unidad es destruida por un misil si ésta se encuentra dentro del radio (**incluido**) de la explosión.

```
>>> disparar((2, 2), 1.5, 'primera_orden.csv', ['GU', 'FU'])
{'GU': 1, 'FU': 1}
>>> disparar((2, 2), 3.5, 'primera_orden.csv', ['SS', 'GU'])
{'GU': 1, 'SS': 0}
```

`primera_orden.csv`

```
4,4;flametrooper;GU
5,0;riot control;GU
```

* *resultado 1er llamado función* `disparar`.

`primera_orden.csv`

```
5,0;riot control;GU
```

* *resultado 2do llamado función* `disparar`.