Programación – Certamen 3 – Semestre 2, 2017

PARALELO ☐☐☐

NOMBRE ☐ ROL ☐☐☐☐☐☐☐☐–☐

1. **[20 %]** Una línea de buses mantiene información de sus viajes en dos archivos, uno llamado `viajes.txt`, en formato: `IDVIAJE#RUTA#HORA#CAPACIDAD`, donde `IDVIAJE` corresponde a un identificador único, `RUTA` el orden en que visita ciudades, `HORA` la hora del viaje en formato `hh:mm` y `CAPACIDAD` que corresponde a la cantidad de asientos del bus. El otro archivo es `compras.txt` en formato: `IDVIAJE:CANTIDAD:RUTCOMPRA`, donde `IDVIAJE` es el identificador único, `CANTIDAD` es el número de asientos comprados y `RUTCOMPRA`, el rut de la persona que los compró.

`viajes.txt`

```
TB2456#Santiago-ElTabo#16:00#30
CB4390#Arica-Pta.Arenas#00:00#20
TT7399#Temuco-Santiago#23:00#25
DF1298#Santiago-ElTabo#16:00#25
...
```

`compras.txt`

```
TB2456:4:11111111-1
CB4390:6:12345678-9
CB4390:3:14375988-4
TT7399:25:15489685-5
...
```

La función `informar(viajes, compras)` recibe los nombres de los dos archivos descritos y escribe la ruta, la hora y cupos disponibles de cada viaje en el archivo `informe.txt`. Para cumplir con esto, utiliza la función `disp(IDVIAJE,cap, compras)`, donde `IDVIAJE` es el identificador del viaje, `cap` es un entero con la capacidad del bus y `compras` el nombre del archivo de compras realizadas. Esta función debe retornar la disponibilidad de asientos en el bus. A continuación se encuentran ambas funciones desordenadas, usted debe ordenarlas e indentarlas para que funcionen en Python:

```
return cap
cap -= int(cant)
v.close()
v=open(compras)
if IDVIAJE == id:
id,cant,rut=linea.strip().split(':')
def disp(IDVIAJE, cap, compras):
for linea in v:
```

```
cap = int(cap)
def informar(viajes, compras):
nline=texto.format(ruta,hora,cupos)
cupos = disp(id,cap,compras)
viaj.close()
infor.close()
infor=open('informe.txt','w')
texto='{0}#{1}#{2}\n'
infor.write(nline)
viaj=open(viajes,'r')
for linea in viaj:
linea=linea.strip().split('#')
id,ruta,hora,cap=linea
```

Nota: Los archivos de ejemplo tienen más líneas.

Programación – Certamen 3 – Semestre 2, 2017

2. **[40 %]** Los tomodatchi son mascotas virtuales que deben ser alimentadas constantemente o sino pueden morir. Como ejemplo, suponga que tiene la siguiente colección de tomodatchis en un archivo llamado `tomodatchi.txt` y los alimentos que pueden comer en el archivo `comidas.txt`. Notar que los dos números contiguos a un tomodatchi o alimento corresponden a la felicidad y satisfacción respectivamente:

`tomodatchi.txt`

```
pedrotchi;5;5
javiertchi;0;3
nicolatchi;3;0
cristophertchi;4;4
viktortchi;2;1
...
```

`comidas.txt`

```
sopaipa;1;2
tocomple;2;1
susushi;1;1
supapajohns;3;3
sumaruchan;1;0
...
```

Nota: Los archivos de ejemplo tienen más líneas.

a.- Desarrolle la función `status(tomo, arch)`, donde `tomo` es un string con el nombre del tomodatchi y `arch` un string con el nombre del archivo de tomodatchis. La función retorna un diccionario con la felicidad y satisfacción de dicho tomodatchi. Si la felicidad o la satisfacción es cero, entonces retorna `False` (tomodatchi muerto). Asuma que el tomodatchi siempre existirá en el archivo. Guíese por el ejemplo.

```
>>status('pedrotchi','tomodatchi.txt')
{'feliz': 5, 'satisfecho': 5}
>>status('javiertchi','tomodatchi.txt')
False
```

b.- Desarrolle la función `alimento(ali, arch)` donde `ali` es un string con el nombre del alimento y `arch` un string con el nombre del archivo de alimentos. La función retorna un diccionario con la felicidad y satisfacción que tiene dicho alimento. Si no lo encuentra retorna `False`. Guíese por el ejemplo.

```
>>alimento('tocomple','comidas.txt')
{'feliz': 2, 'satisfecho': 1}
>>alimento('salchipapa','comidas.txt')
False
```

c.- Desarrolle la función `alimentar(tomo,ali,arch1,arch2)`, donde `tomo` es el nombre del tomodatchi, `ali` es el nombre del alimento, `arch1` es el nombre del archivo con los tomodatchis y `arch2` el nombre del archivo con los alimentos. En el caso de que el tomodatchi esté vivo y que el alimento exista, la función retorna un diccionario con los datos actualizados (sumándole la cantidad de satisfacción y felicidad correspondiente al alimento). Si el tomodatchi está muerto, la función retorna un string `'X.X'`, exista o no el alimento. Finalmente, si el tomodatchi está vivo y el alimento no existe, la función retorna un string `'no existe alimento'`. Asuma que el tomodatchi siempre existirá en el archivo. Guíese por el ejemplo:

```
>>alimentar('pedrotchi','tocomple','tomodatchi.txt','comidas.txt')
{'feliz': 7, 'satisfecho': 6}
>>alimentar('javiertchi','tocomple','tomodatchi.txt','comidas.txt')
X.X
>>alimentar('viktortchi','salchipapa','tomodatchi.txt','comidas.txt')
no existe alimento
```

Programación – Certamen 3 – Semestre 2, 2017

3. **[40 %]** PyLab es un laboratorio que está desarrollando vacunas para 2 tipos de enfermedades. Cada mes **algunos** pacientes son inyectados con vacunas experimentales y se mide la presencia de 3 sustancias en su cuerpo: `BR1, BR2` y `KR`.

En un archivo (`datos.txt`) se guardan los resultados tras probar una nueva vacuna. La información de un paciente estará en una línea del tipo: `CP;G;BR1;BR2;KR`, donde `CP` es el código del paciente, `G` es el género del paciente (1: femenino, 0: masculino) y `BR1, BR2` y `KR` son las 3 sustancias que, cuando están presentes, tienen un 1 y sino un 0. Por ejemplo la paciente `JP87` solo presenta la sustancia `KR`. Notar que entre dos pacientes siempre hay una línea en blanco.
La probabilidad de desarrollar la enfermedad 1 está dada por:

$$\text{Enfermedad–1} = 10 \cdot KR + 30 \cdot BR2 + 4$$

`datos.txt`

```
JP87;1;0;0;1

AY0231;0;1;1;0

DR762399;0;0;0;0

ED235;1;0;0;1

W4350;1;0;1;1

...
```

La probabilidad de desarrollar la enfermedad 2 depende del género y está dada por:

$$\text{Enfermedad–2} = \begin{cases} 15 \cdot \text{BR1} + 65 \cdot \text{BR2} + 3 & \text{si} \quad G == 1, \\ 5 \cdot \text{BR1} + 8 \cdot \text{BR2} + 6 & \text{si} \quad G == 0. \end{cases}$$

Por otro lado se cuenta con el archivo histórico de pacientes en el formato `CP; PE1; PE2`, donde `CP` es el código del paciente, `PE1` es la probabilidad de desarrollar la enfermedad 1 y `PE2` la enfermedad 2. Notar que después de cada `;` existe un espacio en blanco en el archivo histórico.

Escriba la función `actualizar(ad,ah,ah2)`, donde `ad` es el nombre de un archivo de datos tras probar una nueva vacuna (como `datos.txt`), `ah` es el nombre de un archivo de tipo histórico (como `historico.txt`) y `ah2` es el nombre del **nuevo** archivo donde se escribirá la información actualizada considerando los dos archivos anteriormente descritos (el de datos y el histórico).

`historico.txt`

```
JP87; 4 %; 3 %
KL898989; 4 %;  14 %
AY0231; 4 %; 6 %
DR762399; 34 %; 11 %
ED235; 14 %; 3 %
W4350; 14 %; 68 %
...
```

`historico2.txt`

```
JP87; 14 %; 3 %
KL898989; 4 %;  14 %
AY0231; 34 %; 19 %
DR762399; 4 %; 6 %
ED235; 14 %; 3 %
W4350; 44 %; 68 %
...
```

Por ejemplo al ejecutar `actualizar('datos.txt','historico.txt','historico2.txt')`, se generá el archivo `historico2.txt` como se ve en el ejemplo anterior.

El archivo `ah2` será un **nuevo** archivo histórico en donde, si el paciente existía y se probó la vacuna en él, se reemplazan las probabilidades de las enfermedades según el archivo de datos (`ad`). Si el paciente no exisitía, lo agrega en cualquier lugar y finalmente si el paciente existía, pero no se probó la vacuna sobre él, es decir, el paciente no está en el archivo `ad`, lo copia directo en el archivo `ah2`. Guíese por el ejemplo.

Nota: Los archivos de ejemplo tienen más líneas. Recuerde que puede crear todas las funciones auxiliares que estime conveniente.

Programación – Certamen 3 – Semestre 2, 2017 PARALELO

NOMBRE ROL –

2.

Programación – Certamen 3 – Semestre 2, 2017 PARALELO

NOMBRE ROL –

3.