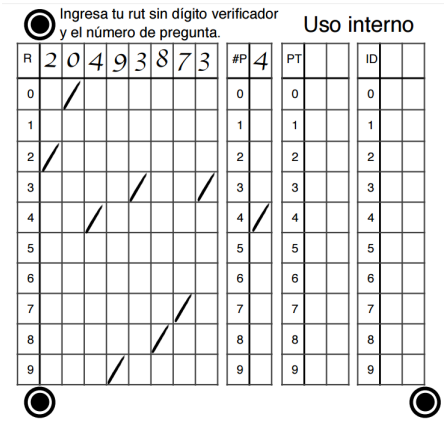
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC1103 – Introducción a la Programación
2 - 2016

Interrogación 2

Instrucciones generales:

- La interrogación consta de tres preguntas con igual ponderación.
- Solo se reciben consultas sobre el enunciado en los primeros 30 minutos.
- La duración de la interrogación es de 2 horas y 30 minutos.
- La última página es un recordatorio; no está permitido utilizar material de apoyo adicional.
- Si el enunciado de la pregunta no lo indica explícitamente, no es necesario validar los ingresos del usuario. Puedes suponer que no hay errores por ingreso de datos.
- Recuerda que esta es una instancia formal de evaluación. Tu respuesta será leída por otros por lo que debes utilizar un lenguaje apropiado. Además, no se aceptarán comentarios que no estén relacionados a la respuesta que se está respondiendo.
- Responde cada pregunta en una hoja separada, marcando tu RUT y el número de pregunta en todas las hojas prestando atención a la dirección del trazo, tal como se muestra en el siguiente ejemplo:



Pregunta 1 (1/3)

La universidad está organizando visitas de colegios a cada una de las carreras que imparte. Para esto, abrió un formulario que le permite a los escolares inscribirse en la visita de su interés. Los nombres de los inscritos y la carrera de su interés están (en orden de llegada) en el archivo `inscritos.txt`. Pero solo hay un cierto número de cupos para la visita a cada carrera, los que se encuentran en el archivo `carreras.txt`.

inscritos.txt

```
Alan Turing,Ingenieria
Eloisa Diaz,Medicina
Gustave Eiffel,Ingenieria
Ada Lovelace,Ingenieria
Larry King,Periodismo
Jonas Salk,Medicina
Grace Hopper,Ingenieria
...
```

carreras.txt

```
Ingenieria 55
Medicina 1
Periodismo 10
Enfermeria 35
College 40
...
```

Se te pide que escribas un programa que, a partir de los archivos anteriores, genere para cada carrera, un archivo (cuyo nombre debe ser el nombre de la carrera en minúsculas, seguido de .txt) con los estudiantes inscritos para la visita. Si hay más interesados que cupos, solo quedan inscritos los primeros que hayan hecho la inscripción. Si no hay ningún interesado en alguna visita, el archivo debe crearse y quedar vacío. A continuación se muestran, a modo de ejemplo, los archivos resultantes para Ingeniería, Medicina y Periodismo.

ingenieria.txt

```
Alan Turing
Gustave Eiffel
Ada Lovelace
Grace Hopper
...
```

medicina.txt

```
Eloisa Diaz
```

periodismo.txt

```
Larry King
...
```

Pregunta 2 (1/3)

En el futuro, es muy posible que existan robots desempeñando labores útiles en espacios públicos. Un problema de computación importante que deberán resolver estos robots es el de calcular un camino entre dos puntos. En este curso no hemos visto como hacer esto; sin embargo, ¡tú ya puedes hacer programas que verifican o reparan caminos!

En esta pregunta debes escribir una función que recibe un mapa y un camino. El mapa se representa con una lista de strings del mismo largo. Cada uno de esos strings está compuesto solo de los siguientes caracteres: `'X'`, que representa un obstáculo; `' '`, que representa un espacio vacío; `'I'`, que representa a una posición inicial (que no es un obstáculo); y `'F'`, que representa a una posición final (que tampoco es un obstáculo). El camino se representa como un string de movimientos separados por comas. Los movimientos posibles son `'arriba'`, `'abajo'`, `'izquierda'`, `'derecha'`.

Por ejemplo, para representar un mapa que se ve así:

X	X	X	X	X	X	X	X
X	X	I	I	X			X
X		F			F		X
X	X	X	X	X	X	X	X

usamos la lista de strings `['XXXXXXXX','XXIIX  X','X F  F X','XXXXXXXX']`. El mapa puede contener múltiples posiciones iniciales y múltiples posiciones finales. El robot puede desplazarse por cualquiera de las posiciones representadas por caracteres, excepto por aquellas marcadas por una `X`. Así, si el robot está ubicado sobre la `I` de más a la derecha y se mueve `abajo` y luego a la `izquierda`, entonces ahora se ubica sobre la `F` que está más hacia la izquierda.

Suponiendo que el mapa siempre contiene obstáculos en los bordes (tal como en el ejemplo):

a) (4 puntos) Programa la función `es_camino(m,c)` que retorna `True` si desde una posición del mapa marcada con `I` es posible seguir el camino `c` llegando a una posición marcada con una `F`. En caso contrario la función retorna `False`.

   Como ejemplo, supón que `mapa` contiene al mapa ejemplo de arriba. Entonces:

   - `es_camino(mapa,'abajo,derecha,derecha')` retorna `True` porque nos hace avanzar desde una `I` (la de más a la derecha) hasta una `F` (la de más a la derecha).
   - `es_camino(mapa,'izquierda,abajo')` retorna `True` porque nos hace avanzar desde una `I` (la de más a la derecha) hasta una `F` (la de más a la izquierda).
   - `es_camino(mapa,'derecha,derecha,abajo')` retorna `False`. De hecho, el camino "choca" con un obstáculo, independientemente de la `I` que usemos para partir.

b) (2 puntos) Programa la función `es_casi_camino(m,c)`, que retorna `True` si es que al cambiar uno y solo uno de los movimientos de `c` por otro, se obtiene un camino que conecta una `I` con una `F`. La función retorna `False` en caso contrario. Por ejemplo, `es_casi_camino(mapa,'abajo,abajo,derecha')` retorna `True` porque el segundo movimiento lo podemos cambiar por `derecha` para convertirlo en un camino. (Note que podríamos cambiar el segundo movimiento por `izquierda` para convertirlo en un camino, también.) Por otro lado, `es_casi_camino(mapa,'derecha,derecha')` retorna `False`, porque no hay ninguna forma de cambiar uno y solo uno de los movimientos de `'derecha,derecha'` por otro para que se convierta en un camino que une a una `I` con una `F`.

Pregunta 3 (1/3)

En esta pregunta se te pide que realices un programa que permita gestionar los monumentos de las distintas ciudades del mundo y la calidad de éstos. En concreto, deberás implementar:

a) (2 puntos) Clase `Monumento`: representa un monumento. Cada `Monumento` tiene un nombre, índice de belleza, índice de accesibilidad e índice de precio. Los índices son números entre 1 y 10 (donde 1 es lo más bajo). Para esta clase, deberás implementar al menos los siguientes métodos:

   - `__init__`: Constructor que recibe los argumentos que permitan crear un objeto de la clase `Monumento` e inicializa los atributos nombre, belleza, accesibilidad y precio
   - `calcular_calidad`: Calcula y retorna la calidad del `Monumento` utilizando la siguiente fórmula:
     $belleza * 50 + accesibilidad * 25 + precio * 25$

- `supera_calidad`: **Llamando al método anterior**, verifica si la calidad del `Monumento` supera cierto límite. Retorna `True` si el Monumento supera la calidad límite y `False` en otro caso.
- `__str__`: Retorna un string con información del `Monumento`. El formato que se debe seguir es `Nombre [Calidad] (belleza-accesibilidad-precio)`. Por ejemplo: `Park Guell [925] (10-9-8)` o `La Moneda [775] (9-6-7)`

b) Clase `Ciudad`: representa una ciudad y sus monumentos. Cada ciudad tiene un nombre y una lista de monumentos. Deberás implementar, al menos, los siguientes métodos:

- `__init__`: Constructor que inicializa los atributos nombre y monumentos. Este último es una lista de objetos de la clase `Monumento`
- `agregar_monumento`: agrega un `Monumento` a la lista de monumentos de la ciudad
- `buscar_anomalias`: busca y retorna una lista con todos los monumentos de la ciudad que tienen una anomalía. Un `Monumento` tiene una anomalía cuando tiene (i) un índice de belleza alto (es decir, muy bello) y un precio con un índice alto (es decir, muy barato) o (ii) un índice de belleza bajo (es decir, muy feo) y un precio con un índice bajo (es decir, muy caro). El índice de belleza y precio se considera bajo cuando éstos son menores o iguales a 4 y alto cuando tienen valores mayores o iguales a 8.
- `ranking`: retorna una lista con los k monumentos ordenados por calidad, de manera decreciente. En caso de empate en la calidad, puedes poner indistintamente uno u otro primero

A continuación, se muestra un ejemplo de programa principal donde se crean objetos de la clase Monumento y Ciudad con sus respectivas llamadas a métodos. Tu código tiene que funcionar con el código principal.
Atención: No debes implementar el código principal de ejemplo. Lo que debes asegurar es que la implementación de las clases `Monumento` y `Ciudad` permita ejecutar lo siguiente:

```
sagrada = Monumento("Sagrada Familia", 10, 10, 4)
guell = Monumento("Park Guell", 10, 9, 8)
montserrat = Monumento("Montserrat", 9, 6, 7)
ramblas = Monumento("Ramblas", 5, 10, 10)
olimpico = Monumento("Estadio Olimpico", 6, 7, 8)
palaumusica = Monumento("Palau de la Musica", 10, 9, 3)
pedrera = Monumento("Pedrera", 9, 10, 6)
campnou = Monumento("Camp Nou", 3, 9, 4)

print("-CALIDAD-")
print(sagrada.calcular_calidad())
print(ramblas.calcular_calidad())

print("-SUPERA CALIDAD-")
print(sagrada.supera_calidad(800))
print(ramblas.supera_calidad(800))

barcelona = Ciudad("Barcelona")
barcelona.agregar_monumento(sagrada)
barcelona.agregar_monumento(guell)
```

```
barcelona.agregar_monumento(montserrat)
barcelona.agregar_monumento(ramblas)
barcelona.agregar_monumento(olimpico)
barcelona.agregar_monumento(palaumusica)
barcelona.agregar_monumento(pedrera)
barcelona.agregar_monumento(campnou)

print("-ANOMALIAS-")
anomalias = barcelona.buscar_anomalias()
for monumento in anomalias:
    print(monumento)

print("-RANKING-")
ranking = barcelona.ranking(5)
for monumento in ranking:
    print(monumento)
```

Al ejecutarse el código anterior, se obtiene la siguiente salida en pantalla:

```
-CALIDAD-
850
750
-SUPERA CALIDAD-
True
False
-ANOMALIAS-
Park Guell[925](10-9-8)
Camp Nou[475](3-9-4)
-RANKING-
Park Guell[925](10-9-8)
Sagrada Familia[850](10-10-4)
Pedrera[850](9-10-6)
Palau de la Musica[800](10-9-3)
Montserrat[775](9-6-7)
```