Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC1103 – Introducción a la Programación
2018 - 1

Interrogación 2

Instrucciones generales:

- **Para resolver la interrogación solo necesitas hasta la sección 11 del Recordatorio de contenidos adjunto.**
- La interrogación consta de 3 preguntas con igual ponderación.
- Solo se reciben consultas sobre el enunciado en los primeros 30 minutos.
- No está permitido utilizar material de apoyo adicional al recordatorio adjunto.
- Todas las preguntas están pensadas para ser resueltas en forma sencilla con lo visto en clases. No es necesario utilizar otros mecanismos más avanzados.
- 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.
- Responde cada pregunta en una hoja separada, marcando tu Número de alumno y el número de pregunta en todas las hojas. Si usas más de una hoja en una pregunta, enumera las hojas. La numeración es por pregunta.
- Puedes utilizar cualquier cosa que te hayan pedido programar en un ítem anterior dentro de las preguntas (incluso si no respondiste ese ítem).
- Tienes dos horas para resolver la prueba desde que se entregan los enunciados.

Pregunta 1

El crucigrama es un juego en el cual uno debe rellenar todas las casillas de una grilla. Las casillas se rellenan según una descripción de la palabra que va en esa fila o columna. Para este problema, considera que tendrás una serie de (cualquier número de) palabras que van horizontalmente y solo una que va verticalmente, como se muestra en la siguiente figura:

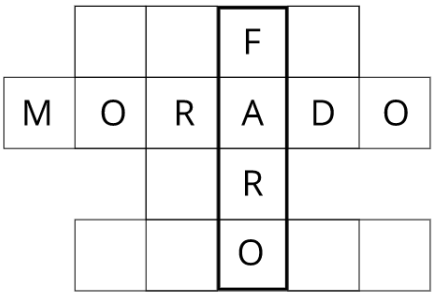

Figura 1: Ejemplo de crucigrama

Tu programa deberá hacer las funciones que permitan rellenar cualquier crucigrama. Para esto, considera que un crucigrama se puede modelar de forma que sea una lista de listas, donde cada lista contiene dos cosas: una lista de letras de la palabra, y un índice, que indica la posición donde intersecta la palabra vertical. Siguiendo el ejemplo anterior, el crucigrama `C` se vería así:

```
C = [
        [['', '', 'F', ''], 2],
        [['M', 'O','R', 'A', 'D', 'O'], 3],
        [['', 'R'], 1],
        [['', '', 'O', '', ''], 2]
]
```

Con esto, debes crear las siguientes funciones:

- **(20 puntos)** `ubicar_horizontal(C,P,I)`: agrega la palabra `P` horizontalmente al crucigrama `C` en la lista de índice `I`, donde `P` es una lista de letras. Se debe verificar que el largo de `P` corresponda con el largo de la palabra horizontal en `I`. Solo se podrá agregar la palabra en el caso que coincida la letra correspondiente a la posición de intersección con la palabra vertical, o en el caso en que ese espacio se encuentre vacío. Si ya había una palabra en la fila `I`, se reemplaza por la nueva palabra en caso de que cumpla las condiciones anteriores. Retorna el crucigrama actualizado (si la palabra no se pudo agregar, retorna el mismo crucigrama anterior). Por ejemplo, para el crucigrama de la Figura 1, `ubicar_horizontal(C,['F','L','O','J','O'],3)` entrega la lista `[[[", ", 'F', "], 2], [['M', 'O', 'R', 'A', 'D', 'O'], 3], [[", 'R'], 1], [['F', 'L', 'O', 'J', 'O'], 2]]`.

- **(20 puntos)** `ubicar_vertical(C,P)`: agrega la palabra `P` de manera vertical al crucigrama `C`, donde `P` es una lista de letras. La palabra **solo se puede agregar si** el largo de `P` corresponde a la altura del crucigrama, y además cada una de las letras de `P` coincide con las letras de las intersecciones con las palabras horizontales, o esos espacios están vacíos. Retorna el crucigrama actualizado (si la palabra no se pudo agregar, retorna el mismo crucigrama anterior). Por ejemplo, para el crucigrama anterior, `ubicar_vertical(C,['C','A','R','O'])` retorna el mismo crucigrama de la Figura 1, `[[[", ", 'F', "], 2], [['M', 'O','R', 'A', 'D', 'O'], 3], [[", 'R'], 1],[[", ", 'O', ", "], 2]]`.

- **(20 puntos)** `imprimir(C)`: Imprime el crucigrama `C` de forma que las celdas se vean alineadas, tal como se muestra en la Figura 1. Si hay una celda que no contenga una letra, debes imprimirla como un guión bajo. Por ejemplo, para el crucigrama de la Figura 1, `imprimir(C)` imprime lo siguiente:

```
0.-   _ _ F _
1.- M O R A D O
2.-     _ R
3.-   _ _ O _ _
```

Pregunta 2

Hackerrank ha tenido fallas de nuevo, por lo que te han pedido ayuda para actualizar los puntajes obtenidos por los alumnos en el último laboratorio. Hasta el momento, la información necesaria está guardada en dos archivos:

- `acumulado_6.txt`: con todos los puntajes obtenidos por los alumnos hasta el laboratorio número 6. Cada línea del archivo tiene el nombre de usuario y su puntaje acumulado, separados por un punto y coma. Nota que el archivo tiene un cantidad arbitraria de alumnos.
- `laboratorio_7.txt`: con los puntajes obtenidos en los ejercicios del laboratorio 7. Cada línea contiene el nombre de usuario, el número del problema resuelto y el puntaje obtenido en él, separados por un punto y coma. Nota que el laboratorio puede tener una cantidad arbitraria de problemas.

Deberás crear un programa que sume los puntajes hasta el laboratorio 6 con los nuevos obtenidos en el laboratorio 7 y los guarde en un archivo llamado `acumulado_7.txt`. Puedes asumir que todos los alumnos que hicieron el laboratorio 7 también han hecho los laboratorios anteriores.
Las siguientes figuras ilustran lo deseado. Por ejemplo, el alumno `iic1181_miguel` tenía 6100 puntos acumulados, y en el laboratorio 7 obtuvo 600 puntos entre los tres ejercicios resueltos. Entonces, en el nuevo archivo se creó una línea con los datos `iic1181_miguel;6700`.

```
iic1181_miguel;6100
iic2181_marcos;2500
iic3181_pablo;6300
iic4181_vale;6400
iic5181_jorge;6500
```

(a) acumulado_6.txt

```
iic1181_miguel;1;100
iic2181_marcos;1;50
iic3181_pablo;1;100
iic4181_vale;1;100
iic5181_jorge;1;100
iic1181_miguel;3;200
iic2181_marcos;3;50
iic3181_pablo;3;200
iic4181_vale;3;200
iic5181_jorge;3;200
iic1181_miguel;5;300
iic3181_pablo;5;300
iic4181_vale;5;300
iic5181_jorge;5;300
```

(b) laboratorio_7.txt

```
iic1181_miguel;6700
iic2181_marcos;2600
iic3181_pablo;6900
iic4181_vale;7000
iic5181_jorge;7100
```

(c) acumulado_7.txt

Pregunta 3

¡Bienvenido al restaurante *DCChurrasco*! El siguiente código usa la programación orientada a objetos y las clases de objetos `Mesa` y `Restaurante` para gestionar el restaurante. En concreto 1) Cuando llega un grupo de clientes, el sistema los sienta en una mesa libre, o les dice que no hay mesas disponibles, 2) Se puede pedir comida para una mesa, indicando el plato y su precio, 3) Al pedir la cuenta el sistema indica el número de platos consumidos por la mesa, el precio total y el precio por persona, en partes iguales, quedando la mesa libre en ese momento, 4) Da la opción de salir del programa. Considera que todas las mesas tienen 4 sillas y que siempre llegarán grupos de al menos 1 persona y a lo más 4 personas.

```
r = Restaurante('DCChurrasco', 2) # restaurante llamado 'DCChurrasco' con 2 mesas vacías

continuar = True
while continuar:
    opcion = int(input('1-Llegan clientes 2-Pedir comida 3-La cuenta 9-Salir: '))

    if opcion == 1:
        cantidad_clientes = int(input('Número de clientes: '))
        numero_mesa = r.llegan_clientes(cantidad_clientes)
        if numero_mesa == -1:
            print('No hay mesas disponibles :(')
        else:
            print('Siéntense en la mesa', numero_mesa)

    elif opcion == 2:
        numero_mesa = int(input('Número de mesa: '))
        plato = input('Plato: ')
        precio = int(input('Precio: '))
        r.mesas[numero_mesa].agregar_plato(plato, precio)

    elif opcion == 3:
        numero_mesa = int(input('Número de mesa: '))
        mesa = r.mesas[numero_mesa]
        print('Pidieron', len(mesa.obtener_platos()), 'platos')
        print('La cuenta son', mesa.obtener_cuenta())
        print('Eso hace', mesa.obtener_cuenta_por_persona(), 'por persona')
        mesa.vaciar_mesa()

    elif opcion == 9:
        continuar = False
        print('Saliendo')
```

Se te pide que, considerando el código anterior, implementes las clases `Mesa` y `Restaurante` con los siguientes métodos:

1) **`Mesa`:**

- **(7 puntos)** `__init__`: Inicializa una mesa vacía. No recibe ningún parámetro.
- **(7 puntos)** `ocupar_mesa`: Establece la mesa como ocupada. Recibe un parámetro: un `int` correspondiente al número de personas que se sientan en la mesa. No retorna nada.
- **(4 puntos)** `esta_vacia`: Retorna `False` si la mesa se encuentra ocupada, `True` en el caso contrario. No recibe ningún parámetro.
- **(4 puntos)** `agregar_plato`: Registra un nuevo plato a los platos pedidos por la mesa. Recibe como parámetros el nombre del plato como un `string` y el precio del plato como un `int`.
- **(4 puntos)** `obtener_cuenta`: Retorna un `int` con la suma de precios de todos los platos pedidos en una mesa. No recibe ningún parámetro.
- **(4 puntos)** `obtener_cuenta_por_persona`: Retorna un `float` con lo que debe pagar cada cliente. Se asume que dividen la cuenta en partes iguales. **Es necesario que se use el método `obtener_cuenta` para hacer este cálculo.** No recibe ningún parámetro.

- **(4 puntos)** `obtener_platos`: Retorna una `lista` de `string` con los platos pedidos por la mesa. No recibe ningún parámetro.
- **(7 puntos)** `vaciar_mesa`: Establece la mesa como vacía, cambiando los atributos necesarios para que pueda ser usada nuevamente por otro grupo de clientes.

2) **`Restaurante:`**

- **(12 puntos)** `__init__`: Inicializa un restaurante. Recibe como parámetro el nombre del restaurante como un `string` y un `int` con el número de mesas vacías (de la clase `Mesa`) que tiene el restaurante al crearse.
- **(7 puntos)** `llegan_clientes`: Recibe un `int` con la cantidad de clientes que llegaron. En caso de que exista una mesa vacía, la mesa queda ocupada con los clientes, y el método retorna un `int` con el número que identifica la mesa. Considera que las mesas de un restaurante se identifican con un número del 0 al número de mesas - 1. En caso de que haya más de una mesa vacía, el sistema puede elegir cualquiera. En caso de que no haya ninguna mesa vacía, el método simplemente retorna -1.