Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Segundo Semestre de 2014

IIC 1103 - Introducción a la Programación

Laboratorio 10

Objetivo General

En este laboratorio reforzaremos los conceptos vistos en clases relacionados a la Programación Orientada a Objetos y al uso de archivos. Para llevar a cabo los ejercicios, deberás modelar e implementar clases, sus atributos y métodos. También deberás manejar archivos guardados en el almacenamiento secundario (probablemente el disco duro) de un computador.

Enunciado

1. La memoria principal del computador es secuencial (una lista formada de números binarios de 32 ó 64 bits). Esto implica que toda variable que definimos se debe guardar como una secuencia (o lista) de números. En particular, para codificar una matriz existen dos formatos: Row-Mayor Order y Column-Mayor Order. La codificación de una matriz, Row-Mayor genera una secuencia que contiene las filas de la matriz escritas una tras otra, mientras que el segundo genera una secuencia que contiene las columnas de la matriz, una tras otra. Además, ambos incluyen las dimensiones de la matriz al inicio de la secuencia. Por ejemplo, para codificar la matriz $A_{2\times 3}$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

en Row-Mayor order se genera la secuencia `"2 3 1 2 3 4 5 6"`, mientras que para codificarla en Column-Mayor order se genera la secuencia `"2 3 1 4 2 5 3 6"`. Cabe notar que los números 2 y 3 que aparecen al principio de ambas codificaciones representan las dimensiones de la matriz, es decir, 2×3.

En esta pregunta deberás crear un programa que lea una matriz de un archivo y luego muestre en consola dicha matriz codificada de las dos formas antes descritas. En el archivo los valores de la matriz serán números **enteros** separados por un espacio. Por ejemplo, un archivo que contiene a la matriz A anterirmente descrita se vería de la siguiente forma:

```
1 2 3
4 5 6
```

Tu programa debe recibir como entrada la ruta (ya sea absoluta o relativa) al archivo que contiene los valores de una matriz. Por ejemplo, si el archivo anterior se llama *matriz.txt* y está ubicado en la misma carpeta que tu programa, el input será igual a `"matriz.txt"`.

La salida de tu programa debe contener dos líneas: la primera es la codificación Row-Mayor y la segunda la codificación Column-Mayor de la matriz que se encuentra en el archivo indicado en la entrada del programa. En cada codificación los valores deben estar separados con espacio. Por ejemplo, para el caso anterior la salida de tu programa debería ser:

```
2 3 1 2 3 4 5 6
2 3 1 4 2 5 3 6
```

Ten en cuenta que este laboratorio lo revisa un computador, por lo que tu respuesta será mal calificada si hay un espacio al principio al final de alguna línea.

2. Un restaurant de comida rápida tiene guardado un listado de las direcciones de sus clientes. Para esto, utiliza un archivo en el que cada línea contiene la dirección de un cliente y el número que identifica a dicho cliente separados por coma y sin espacios, tal como se muestra a continuación:

```
direccion_cliente_1,numero_cliente_1
direccion_cliente_2,numero_cliente_2
...
direccion_cliente_n,numero_cliente_n
```

En otro archivo están guardados los precios de los platos que ofrece el restaurant. En cada línea de este archivo se encuentra el número que identifica a un plato seguido de el precio de dicho plato plato separados por coma y sin espacios, tal como se muestra a continuación:

```
numero_plato_1,precio_1
numero_plato_2,precio_2
...
numero_plato_n,precio_n
```

En este restaurant los clientes suelen hacer muchos pedidos, por lo que tu trabajo será consolidar la información que ingresa al restaurant para calcular el total que debe pagar cada persona que compró algo. Como entrada tu programa recibirá primero el nombre del archivo con las direcciones y los clientes, y luego el nombre del archivo con los platos y sus precios. Luego recibirá una cantidad arbitraria de órdenes de clientes. Cada orden de un cliente viene representada por una línea en la que se encuentra el número de un cliente y el número de un plato, separados por coma y sin espacios. Debes terminar de recibir entradas una vez que se ingrese una línea vacía. A continuación se muestra un ejemplo de órdenes ingresadas:

```
numero_cliente_1,numero_plato_1
numero_cliente_2,numero_plato_2
...
numero_cliente_M,numero_plato_M
<Linea en blanco>
```

Por supuesto que un cliente puede aparecer más de una vez (puede ordenar más de un plato), y los clientes no necesariamente van a venir ordenados, ya que hay más de una persona atendiendo pedidos.
Como output, tu programa debe indicar el precio total que deberá pagar cada cliente. Para esto, debes imprimir una línea por cliente que haya comprado algo, la cual debe contener el número del cliente, la dirección del cliente y el precio total a pagar separados por coma y sin espacios tal como se muestra a continuación:

```
numero_cliente_1,direccion_cliente_1,total_a_pagar1
...
numero_cliente_N,direccion_cliente_N,total_a_pagarN
```

Además, deberás imprimir estas líneas ordenadas de acuerdo al número de cliente. Por ejemplo, supongamos que el contenido de *direcciones.txt* es:

```
Av. Providencia 1102,1
Av. Los Leones 1204,2
```

y que el contenido de *precios.txt* es:

```
0,5600
1,1500
2,3000
```

Entonces, el siguiente sería un ejemplo de diálogo para este programa: Puedes suponer que

Entrada	Salida
1,0	
2,1	
2,2	
1,2	
	1,Av. Providencia 1102,8600
	2,Av. Los Leones 1204,4500

los nombres de los platos y las direcciones de los clientes no contienen comas, y que los archivos indicados en la entrada tendrán siempre el formato correcto.

3. Un temerario guerrero se ha introducido a la mazmorra del DCC en busca de un tesoro. Para llegar a él, deberá moverse por la mazmorra evitando chocar con muros y bestias. Deberás realizar un programa que reciba como input la ruta de un archivo que contiene el mapa de la mazmorra y luego un listado con los movimientos que realiza el guerrero.

   El mapa de la mazmorra posee la siguiente simbología:

   - El caracter espacio representa un pasillo o posición libre.
   - El caracter `"x"` representa un muro.
   - El caracter `"t"` representa al tesoro.
   - El caracter `"b"` representa una bestia.
   - El caracter `"g"` representa al guerrero.

   Puedes suponer que los bordes de una mazmorra siempre tienen muros. Para construir un archivo con el mapa de una mazmorra se utiliza una matriz con la simbología descrita arriba. Por ejemplo, el siguiente es un archivo que representa una mazmorra:

```
xxxxxxxxxx
x t b b  x
xxx  b   x
x g xx   x
xxxxxxxxxx
```

   Tu programa recibirá como primera entrada la ruta de un archivo en el que está guardado el mapa de una mazmorra. Luego recibirá como input una línea que representa movimientos que hará el guerrero. Esta línea contendrá unicamente los caracteres N, S, E y O, que equivalen a dar un paso en la dirección norte, sur, este y oeste, respectivamente. Por ejemplo, la entrada `ENNO` significa que el guerrero avanza un paso a la derecha, dos pasos hacia arriba y uno a la izquierda. Deberás ejecutar uno a uno cada movimiento del

guerrero hasta que llegue al tesoro, choque con un muro, lo mate una bestia, o se terminen los movimientos indicados en la entrada. Dependiendo de cuál de las opciones anteriores ocurra, tu programa deberá imprimir uno de los siguientes mensajes:

```
# Si el guerrero choca con un muro
print("El guerrero se ha desorientado, intente en otro momento.")
# Si el guerrero encuentra una bestia
print("El guerrero ha muerto a manos de la bestia.")
# Si el guerrero encuentra el tesoro
print("Yei.")
# Si se terminan los movimientos sin llegar al tesoro
print("El guerrero abandona la mazmorra sin suerte alguna.")
```

Por ejemplo, si el archivo que contiene la mazmorra que se muestra arriba es `mazmorra.txt`, los siguientes serían tres diálogos posibles para tu programa:

Caso	Entrada	Salida
1	mazmorra.txt ENNO	Yei.
2	mazmorra.txt EEN	El guerrero se ha desorientado, intente en otro momento.
3	mazmorra.txt ENNE	El guerrero ha muerto a manos de la bestia.

Recuerda mostrar los mensajes tal cual se muestran arriba, pues si falta cualquier símbolo de puntuación u olvidas una mayúscula o tilde tu respuesta será mal calificada.

4. En esta pregunta deberás ayudar a coordinar y ordenar la información de una empresa de autobuses. Más específicamente, deberás hacer un programa que guarde información respecto de en qué autobús viaja cada pasajero. Además tu programa deberá ser capaz de responder preguntas respecto de la información guardada.
   Para que el usuario pueda especificar la información necesaria, tu programa recibirá como input lo siguiente: En la primera línea habrá dos números B y P separados por un espacio, los cuales representan la cantidad de buses y la cantidad de pasajeros respectivamente. Las siguientes B líneas tienen cada una dos números enteros separados por un espacio, el primero corresponde al número identificador de un bus y el segundo es la cantidad de asientos que tiene dicho bus. Cada una de las siguientes P líneas también tienen dos números separados por un espacio. El primero es el identificador de un pasajero y el segundo el identificador del bus al que subirá dicho pasajero.
   Una vez que está guardada toda la información anterior, el usuario podrá hacer una cantidad arbitraria de consultas sobre la información guardada. Cada consulta corresponde a una línea que contiene una letra y un número, separados por un espacio. La letra puede ser *b* ó *p*, dependiendo de si se quiere consultar por un bus o por un pasajero, respectivamente. El número será el identificador del bus o pasajero por el cuál se quiere consultar. Para responder una consulta sobre un bus, tu programa debe imprimir una línea con los identificadores de los pasajeros que subieron a dicho bus separados por un espacio y ordenados de menor a mayor. Para responder una consulta sobre un pasajero simplemente se debe imprimir el id del bus al cual subió dicho pasajero. Tu programa debe detenerse cuando lea una línea en blanco. En caso de que se sobrepase la capacidad de un bus, debe imprimir "Bus <id bus> sobrepasado." y terminar la ejecución del programa.

   A continuación se muestran dos posibles diálogos que ejemplifican cómo debiese funcionar tu programa:

Caso	Entrada	Salida
1	2 3 0 2 1 8 0 0 1 1 2 0 p 0 b 1 b 0	 0 1 1 2
2	2 7 0 3 1 5 0 0 1 0 2 1 3 0 4 0	 Bus 0 sobrepasado.