Ejercicios de segundo parcialito
Taller 09/05

Del segundo parcialito, primer cuatrimestre de 2013

1. Escribir la clase Polinomio, utilizando una lista para guardar los coeficientes del mismo. Implementar los métodos:

- init : Constructor. Recibe una lista con los coeficientes.
- str : Devuelve una representación de cadena del polinomio, por ejemplo: "3xˆ3 - x + 1".
- eq : Recibe otro Polinomio y devuelve True si son iguales o False si no.
- derivar: Devuelve un nuevo Polinomio que es la derivada del actual.

Del segundo recuperatorio del segundo parcialito, primer cuatrimestre de 2013

2. Escribir un programa que abra un archivo separado por comas y le pida al usuario dos números. El programa debe escribir en otro archivo el contenido del primero con las columnas del archivo CSV intercambiadas según los números del usuario. Por ejemplo, si una línea del archivo es "lunes, martes,miércoles,jueves" y los números del usuario son 1 y 3, el archivo de salida será "miércoles, martes,lunes, jueves" (las columnas 1 y 3 están intercambiadas)

Del tercer parcialito, primer cuatrimestre de 2012

3. Implementar una clase Colectivo con los siguientes métodos:

- Un constructor, que crea un Colectivo vacío.
- Un método subir, que dado un destino (en forma de cadena) y un objeto de la clase Persona, lo agrega al Colectivo.
- Un método bajar, que dado un destino, saca del Colectivo a las Personas que tienen dicho destino asignado, devolviéndolas en una lista.

Del tercer parcialito, primer cuatrimestre de 2010

4. Escribir una función que reciba un nombre de un archivo con datos y un nombre de un archivo para guardar errores (los errores se agregan al final). El archivo con datos tiene el siguiente formato:

año , mes , vendedor , cliente , monto

La función debe devolver un diccionario con los vendedores y los montos totales por vendedor como valor. Cuando una línea no sea válida, la ejecución no se puede detener, pero debe guardar la línea en el archivo de errores.

5. Escribir una función que reciba un diccionario con `vendedor - monto`, otro diccionario con `vendedor - comisión`, y un archivo en el cual grabar los datos. La función debe validar que todos los vendedores del primer diccionario se encuentren en el segundo diccionario, y de no ser así levantar una excepción; calcular el monto que le corresponde a cada uno ($monto * comision$), y guardarlo en el archivo de salida como $< vendedor >:< monto >$.