Ejercitación № 5: Cadenas, Listas y Tuplas

Cátedra Programación II Septiembre 2016

1. Cadenas

1) Escriba un programa que tenga una función que tome una cadena y muestre cada caracter que la forma del último al inicial.

2) Escriba un programa que contenga a la función `contar(l, x)` que cuente cuántas veces aparece un carácter `l` dado en una cadena `x`.

3) Escriba un programa que cuente cúantas veces aparecen cada una de las vocales en una cadena. No importa si la vocal aparece en mayúscula o en minúscula.

4) Escriba un programa que contenga la función que reciba como parámetro una cadena de palabras separadas por espacios y devuelva, como resultado, cuántas palabras de más de cinco letras tiene la cadena dada.

2. Listas

1) Escriba una función que tome una lista de números y devuelva la suma acumulada, es decir, una nueva lista donde el primer elemento es el mismo, el segundo elemento es la suma del primero con el segundo, el tercer elemento es la suma del resultado anterior con el siguiente elemento y así sucesivamente. Por ejemplo, la suma acumulada de $[1, 2, 3]$ es $[1, 3, 6]$.

2) Escriba una función llamada `elimina` que tome una lista y elimine el primer y último elemento de la lista y cree una nueva lista con los elementos que no fueron eliminados.

3) Escriba una función `ordenada` que tome una lista como parámetro y devuelva True si la lista está ordenada en orden ascendente y devuelva False en caso contrario. Por ejemplo, `ordenada([1, 2, 3])` retorna $True$ y `ordenada([b, a])` retorna $False$

4) Escriba una función llamada `duplicado` que tome una lista y devuelva True si tiene algún elemento duplicado. La función no debe modificar la lista.

5) Escriba una función llamada `eliminaDuplicados` que tome una lista y devuelva una nueva lista con los elementos únicos de la lista original. No tienen porque estar en el mismo orden.

6) Para comprobar si una palabra está en una lista se puede utilizar el operador **in**, pero sería una búsqueda lenta, ya que busca a través de las palabras según el orden que están en la lista. Si la lista almacena las palabras en orden alfabético, podemos acelerar las

cosas con una búsqueda dicotómica (también conocida como búsqueda binaria), que es similar a lo que hacés cuando buscas una palabra en el diccionario. Comenzamos por el centro y comprobamos si la palabra que buscamos está antes o después del centro. Si está antes, se busca solo en la primera mitad, si está después se busca en la otra mitad de la lista. Con esto reduciremos el tiempo de búsqueda.
Se pide que implemente la función `busquedaDicotomica` que toma una lista de palabras ordenadas alfabeticamente y, una palabra a buscar y, retorna si la palabra está en la lista o no.

3. Tuplas

1) Cartas como tuplas

   a) Proponga una representación con tuplas para las cartas de la baraja francesa.

   b) Escriba una función poker que reciba cinco cartas de la baraja francesa e informe (devuelva el valor lógico correspondiente) si esas cartas forman o no un poker (es decir que hay 4 cartas con el mismo número).

2) El tiempo como tuplas

   a) Proponer una representación con tuplas para representar el tiempo.

   b) Escribir una función `sumaTiempo` que reciba dos tiempos dados y devuelva su suma

3) Escribir una función `diaSiguienteE` que dada una fecha expresada como la terna *(Día, Mes, Año)* (donde Día, Mes y Año son números enteros) calcule el día siguiente al dado, en el mismo formato.

4) Escriba una función `diaSiguienteT` que dada una fecha expresada como la terna *(Día, Mes, Año)* (donde Día y Año son números enteros, y Mes es el texto Ene, Feb, ..., Dic, según corresponda) calcule el día siguiente al dado, en el mismo formato.

5) Dominó

   a) Escriba una función que indique si dos fichas de dominó encajan o no. Las fichas son recibidas en dos tuplas, por ejemplo: $(3, 4)$ y $(5, 4)$.

   b) Escriba una función que indique si dos fichas de dominó encajan o no. Las fichas son recibidas en una cadena, por ejemplo: `[3-4 2-5]`. Nota: utilizar la función split de las cadenas.