Gobstones

Introducción a la Programación - Práctica 11

Listas

CONSEJOS:

- Leer el enunciado en su totalidad y pensar en la forma de resolverlo **ANTES** de empezar a escribir código.
- Si un ejercicio no sale, se puede dejar para después y continuar con los ejercicios que siguen.
- Los ejercicios están pensados para ser hechos después de haber mirado la teórica correspondiente, y esta después de las actividades de indagación.
- Los ejercicios están tomados de las guías prácticas utilizadas en la materia de Introducción a la Programación de la Universidad Nacional de Quilmes por Pablo Ernesto “Fidel” Martínez López y su equipo, Introducción a la Programación de la Universidad Nacional de Hurlingham de Alan Rodas Bonjour y su equipo, de ejercicios y actividades realizadas por Federico Aloi y Miguel Miloro, a su vez basada en las guías Ejercicios de Introducción a la Programación del CIU General Belgrano, elaboradas por Carlos Lombardi y Alfredo Sanzo, y Fundamentos de la Programación del Proyecto Mumuki. Agradecemos a todos los que nos ayudaron con su inspiración.
- **Realizar en papel los ejercicios que así lo indiquen.**
- **Sí un ejercicio indica BIBLIOTECA significa que será útil para la realización de futuros ejercicios tanto en esta guía como en las siguientes, pudiendo ser utilizado sin tener que volver a definirlo. Es útil mantener registro de dichos procedimientos en su carpeta.**

Última actualización: 5 de Noviembre 2021

INTRODUCCIÓN A LISTAS:

1. Singular

   Escribir la función **`singularCon_`** que dado un elemento describe la lista que tiene únicamente el elemento dado.

2. El segundo

   Escribir la función **`segundoDe_`** que dada una lista, describe el segundo elemento de la lista. Puede asumir que la lista tiene al menos dos elementos.

3. Es singular

   BIBLIOTECA Escribir la función **`esSingular_`** que dada una lista indica sí la misma es singular, es decir, sí tiene únicamente un elemento. Notar que la función debe ser total.
   AYUDA: considerar utilizar **`esVacía`** en combinación con otras operaciones primitivas para arribar a una solución.

4. Las cartas en la mano

   Cuando jugamos a las cartas, las cartas que un jugador sostiene en la mano pueden ser pensadas como una lista, en donde el orden es aquel en el que el jugador va a lanzar las cartas a la mesa. Llamamos "mano" a esta lista de cartas. También podemos pensar en un mazo como una lista de cartas, ordenadas de forma tal que la primera carta de la lista es el tope del mazo. Vamos a llamar "mazo" a esta lista. El tipo de los elementos de estas listas es **`Carta`**, que ya vimos y usamos en la práctica anterior. Con esto en mente se pide que escriba algunas funciones para un juego de cartas genérico.

   a. **`primerCartaDeLaMano_`**, que dada una "mano" describa la primera carta a jugar.

   b. **`segundaCartaDeLaMano_`**, que dada una "mano", describa la segunda carta a jugar.

   c. **`tercerCartaDeLaMano_`**, que dada una "mano", describa la tercera carta a jugar.

   d. **`laMano_LuegoDeRobarUnaCartaDe_`**, que dada una "mano" y un "mazo" describe la mano resultante luego de robar la primera carta del mazo y agregarla a la mano del jugador.

   e. **`laMano_LuegoDeJugarUnaCarta`**, que dada una "mano", describe la mano resultante luego de jugar la próxima carta de la mano.

   f. **`laMano_LuegoDeJugarLaSegundaCarta`**, que dada una "mano", describe la mano resultante luego de jugar exclusivamente la segunda de las cartas.

5. La tira de cartas

   Para un juego particular se necesita armar "tiras" de cartas. Una tira de cartas es una secuencia de cartas en orden creciente, es decir, una lista con un orden particular. Para agregar una carta a una tira (al comienzo de la misma), esta debe ser de diferente palo y menor a la primera carta de la tira. Se pide entonces que escriba las siguientes funciones:

   a. **`primerasTresCartasDeLaTira_`**, que dada una "tira", describe una lista con las primeras 3 cartas de la misma.

b. **laMano_TieneJugadaParaAgregarALaTira_**, que dada una mano y una tira, indique si la primera carta que puede jugar el jugador puede ser agregada en la tira o no.

c. **laTira_DespuésDeJugar_**, que dada una tira de cartas y una carta que cumple las condiciones para ser agregada en la misma, describa la tira que resulta de jugar esa carta en esa tira.

2. Pistas de carreras

Queremos modelar unas pistas de carreras de un juego, para lo cual vamos a utilizar una lista de direcciones. La pista podemos pensarla entonces como dividida en tramos, donde cada tramo va a es la conjunción de dos elementos de la lista. Por ejemplo, la pista [Norte, Norte, Este, Este] es una pista con tres tramos, el primero es una recta, Norte-Norte, el segundo es una curva a la derecha, Norte-Este, y el tercero otra recta al Este, Este-Este.

Vamos a decir que el "siguiente tramo" es el primero de los tramos de una pista. Pedimos entonces que escriba las siguientes funciones:

a. **sigueRectaEn_**, que dada una pista que tiene un siguiente tramo, indique si el siguiente tramo es una recta.

b. **sigueCurvaADerechaEn_**, que dada una pista que tiene un siguiente tramo, indica sí el siguiente tramo es una curva a la derecha.

c. **sigueCurvaAIzquierdaEn_**, que dada una pista que tiene un siguiente tramo, indica sí el siguiente tramo es una curva a la izquierda.

d. **hayUnSiguienteTramoEn_**, que dada una pista indique sí la misma tiene un siguiente tramo. La función debe ser total.

e. **sigueUnaCurvaEn_**, que dada una pista indique sí el siguiente tramo es una curva. La función debe ser total.

f. **siguienteTramoEn_EsVálido**, que dada una pista que tiene un siguiente tramo, indique sí el mismo es válido. Un tramo válido es una recta o una curva.

CONSTRUYENDO LISTAS:

6. Todos igualitos

BIBLIOTECA Escribir la función **listaCon_Repetido_Veces** que dado un elemento de cualquier tipo y un número, describe una lista con que tiene tantos elementos como el número dado, en donde cada elemento es el dado.
Ejemplo: `listaCon_Repetido_Veces(Rojo, 5)` describe la lista `[Rojo, Rojo, Rojo, Rojo]`.

7. Todos seguiditos

BIBLIOTECA Escribir la función **listaDesde_Hasta_** que dados dos enumerativos (por ejemplo, números), en donde el primero es más chico que el segundo, describe la lista que va del primero de los elementos, al segundo de los elementos, pasando por cada elemento intermedio.
Ejemplo: `listaDesde_Hasta_(3, 7)` describe la lista `[3,4,5,6,7]`, y `listaDesde_Hasta_(Azul, Rojo)` describe la lista `[Azul, Negro, Rojo]`.

RECORRIENDO LISTAS Y ACUMULANDO:

8. ¿Cuántos son?

BIBLIOTECA Escribir la función **`longitudDe_`**, que dada una lista cualquiera, describa la cantidad de elementos de la misma.
Ejemplo: `longitudDe_([Azul,Azul,Verde,Rojo])` describe 4.

9. ¿Y cuánto suman?

BIBLIOTECA Escribir la función **`sumatoriaDe_`**, que dada una lista de números, describa la suma de todos los elementos de la misma.
Ejemplo: `sumatoriaDe_([1,10,15,7,9])` describe el número 42, porque `1+10+15+7+9` es igual a 42.

10. ¿Y cuánto multiplican?

BIBLIOTECA Escribir la función **`productoriaDe_`**, que dada una lista de números, describa el producto de todos los elementos de la misma.
Ejemplo: `productoriaDe_([1,5,7,9])` describe el número 315, porque `1*5*7*9` es igual a 315.

11. Puntaje de cartas

Escribir la función **`puntosEn_`**, que dada una mano (lista de Carta) describe la cantidad de puntos que el jugador tiene en su mano, donde:

- Cada carta otorga tantos puntos como el número de la carta
- Las cartas negras otorgan todas 20 puntos.

RECORRIENDO LISTAS Y CONSTRUYENDO OTRAS:

12. Aumentados

Escribir la función **`elementosDe_aumentadosEn_`**, que dada una lista de números, y un número que representa una cantidad a aumentar, describe la lista en donde cada elemento fue aumentado en la cantidad dada[1].
Ejemplo: `elementosDe_aumentadosEn_([1,5,7,9], 3)` describe la lista `[4,9,10,12]`.

13. Opuestos

BIBLIOTECA Escribir la función **`opuestasDe_`**, que dada una lista de direcciones describe la lista donde cada elemento es la dirección opuesta a la misma.
Ejemplo: `opuestosEn_([Norte, Este, Este])` describe la lista `[Sur, Oeste, Oeste]`.

---

[1] Cuando decimos “donde cada elemento...”, hacemos un pequeño abuso del lenguaje, para referirnos a que hay un vínculo entre la lista dada como parámetro (lista original) y la que la función va a describir (lista resultado), para cada uno de sus elementos, en cada una de las posiciones. Por ejemplo, sí decimos “donde cada elemento está aumentado en uno”, podemos decir que la lista resultado y la lista original tienen que tener la misma cantidad de elementos, y para la cada ubicación de la lista resultado, el elemento allí debe corresponderse al elemento en la misma posición en la lista original, pero sumado en uno, por ej. sí la lista original es [1, 2, 3] la lista resultado deberá ser [2, 3, 4].

14. Siguientes

BIBLIOTECA Escribir la función **`siguientesDe_`**, que dada una lista de elementos de algún tipo enumerativo primitivo (Color, Dirección, Número, Booleano) describe la lista donde cada elemento es el siguiente del dado.
Ejemplo: `siguientesDe_([Norte, Este, Este])` describe la lista `[Este, Sur, Sur]` y `siguientesDe_([Rojo, Azul, Verde, Azul])` describe la lista `[Verde, Negro, Azul, Negro]`

15. El toque de Midas

Escribir la función **`transformadasTodasEnOro_`**, que dada una lista de cartas describe la lista donde cada elemento es la misma carta, pero de Oro, por ej. Sí la lista tiene el 7 de Basto y el 3 de Espadas, se debe devolver la lista con el 7 de Oro y el 3 de Oro.

16. Reverso

BIBLIOTECA Escribir la función **`reversoDe_`**, que dada una lista describe su reverso.
Ejemplo: `reversoDe_([Rojo, Azul, Negro, Verde])` describe la lista `[Verde, Negro, Azul, Rojo]`.

FILTRANDO LISTAS:

17. Solo los pares

Escribir la función **`paresDe_`**, que dada una lista de números describe una lista solo con los pares de dicha lista[2].
Ejemplo: `paresDe_([1, 4, 2, 3, 7, 8, 9])` describe la lista `[4, 2, 8]`.

18. Solo los impares

Escribir la función **`imparesDe_`**, que dada una lista de números describe una lista solo con los impares de dicha lista.
Ejemplo: `imparesDe_([1, 4, 2, 3, 7, 8, 9])` describe la lista `[1, 3, 7, 9]`.

19. Sacando elementos

BIBLIOTECA Escribir la función **`elementosDe_SinAparicionesDe_`**, que dada una lista de elementos de cualquier tipo y un elemento, describe una lista solo con los elementos de la lista original que no son el elemento dado, es decir, la lista donde se quitaron todas las apariciones del elemento dado.
Ejemplo: `elementoDe_SinAparicionesDe_([1, 4, 1, 3, 1, 8, 9], 1)` describe la lista `[4, 3, 8, 9]`.

20. Sacando elementos

Escribir la función **`soloLasFigurasDe_`**, que dada una lista de cartas, describe una lista solo con aquellas cartas que son figuras.

---

[2] De la misma forma que antes, cuando decimos “solo con...” nos referimos a que la lista resultado debe tener solo algunos de los elementos que estaban en la lista original, pero otros no, y donde el orden en el que aparecen esos elementos es el mismo en el que aparecían en la lista original.

BUSCANDO EN LISTAS:

21. ¿Está o no está?

BIBLIOTECA Escribir la función **`contiene_A_`**, que dada una lista de elementos de cualquier tipo, y un elemento, indica si el elemento dado está en la lista.
Ejemplo: `contiene_A_([1, 4, 2, 3, 7, 8, 9], 3)` describe verdadero, ya que 3 está en la lista, mientras que `contiene_A_([1, 4, 2, 3, 7, 8, 9], 6)` describe falso, ya que 6 no está en la lista.

22. Dónde está

BIBLIOTECA Escribir la función **`indiceEn_De_`**, que dada una lista de elementos de cualquier tipo, y un elemento, describe la primera ubicación de la lista en la que se encuentra dicho elemento. Puede asumir que el elemento está en la lista.
Ejemplo: `indiceEn_De_([1, 4, 2, 3, 7, 3, 9], 3)` describe 4, ya que 3 está en la cuarta posición de la lista, mientras que `contiene_A_([1, 4, 2, 3, 7, 3, 9], 4)` describe 2, ya que 4 está en la segunda posición de la lista.

23. Ni grande, ni chico

Escribir la función **`hayAlgunoDe_Entre_Y_`**, que dada una lista de números y dos números que describen la cota mínima y máxima respectivamente, indica sí algún elemento de la lista dada está entre la cota dada, inclusive.
Ejemplo: `hayAlgunoDe_Entre_Y_([1, 4, 8], 3, 5)` describe verdadero, ya que 4 está entre 3 y 5.

24. Se carteó

Escribir la función **`hayAlgúnAsEn_`**, que dada una lista de cartas, indica sí en dicha lista hay alguna que sea un as (un 1).

MÍNIMOS Y MÁXIMOS:

25. El más chico de la lista

BIBLIOTECA Escribir la función **`minimoEn_`**, que dada una lista de números describe el número más chico en la lista.
Ejemplo: `minimoEn_([5, 4, 2, 3, 7, 8, 9])` describe 2.

26. El más grande de la lista

BIBLIOTECA Escribir la función **`máximoEn_`**, que dada una lista de números describe el número más grande en la lista.
Ejemplo: `máximoEn_([5, 4, 2, 3, 7, 8, 9])` describe 9.

27. La mejor carta

Escribir la función **`laMejorCartaEn_ParaTira_`**, que dada una lista de cartas, describe la mejor posible para sumar a una tira. Recordemos que para sumar a una tira se debe jugar una carta que sea menor a la primera de la tira y de distinto palo. Por otro lado, el concepto de mejor está dado por el

palo de la carta jugada, donde Oro es la mejor posible, seguido de Copas, luego Espadas y finalmente Bástos.

OTROS EJERCICIOS COMPLEJOS SOBRE LISTAS:

28. Todos hasta

BIBLIOTECA Escribir la función **`elementosEn_Hasta_`**, que dada una lista de elementos y un índice (posición de la lista), describe la lista de todos los elementos en la original que están antes de dicho índice.
Ejemplo: `elementosEn_Hasta_([5, 4, 2, 3, 7, 8, 9], 4)` describe la lista `[5, 4, 2]`, es decir, todos los elementos hasta la posición 4.

29. Todos desde

BIBLIOTECA Escribir la función **`elementosEn_Desde_`**, que dada una lista de elementos y un índice (posición de la lista), describe la lista de todos los elementos en la original que están después de dicho índice.
Ejemplo: `elementosEn_Desde_([5, 4, 2, 3, 7, 8, 9], 4)` describe la lista `[7, 8, 9]`, es decir, todos los elementos hasta la posición 4.

30. Sin elemento

BIBLIOTECA Escribir la función **`elementosEn_SinPrimeraAparicióNDe_`**, que dada una lista de elementos y un elemento que puede o no estar en la lista, describe la lista sin la primera aparición de dicho elemento.
Ejemplo: `elementosEn_SinPrimeraAparicióNDe_([5, 4, 2, 3, 4, 8, 9], 4)` describe la lista `[5, 2, 3, 4, 8, 9]`, es decir, todos los elementos menos el primer 4.

31. Sacando las copias

BIBLIOTECA Escribir la función **`sinDuplicados_`**, que dada una lista de elementos, describe la lista sin duplicados.
Ejemplo: `sinDuplicados_([7, 5, 2, 2, 7, 8, 5])` describe la lista `[7, 5, 2, 8]`, o `[2, 7, 8, 5]`, dependiendo de la estrategia que elija para solucionar el problema.

32. Construir las funciones

a.

b. **laLista_Clonada_Veces**, que dados una lista de elementos y un número, describa la lista resultante de clonar la lista dada tantas veces como se indica.
Por ejemplo, **laLista_Clonada_Veces([Rojo,Azul,Verde], 3)** retorna **[Rojo,Azul,Verde,Rojo,Azul,Verde,Rojo,Azul,Verde]**

c. **losElementosDe_Clonados_Veces**, que dados una lista de elementos y un número, describa la lista que contenga los elementos dados en el orden de dicha lista, pero repetidos la *cantidad* de veces indicada.
Por ejemplo, **losElementosDe_Clonados_Veces([Rojo,Azul,Verde], 3)** retorna **[Rojo,Rojo,Rojo,Azul,Azul,Azul,Verde,Verde,Verde]**.

   d. La solución anterior, ¿fue construida reutilizando **repetición_VecesDe_**? Si no es así, resolverla nuevamente utilizando esa función.

33. Construir la función **lista_HomologadaPorDebajoDe_A_**, que dada una lista de números, y dos números **umbral** y **default**, describa una lista de números que está basada en la lista dada de la siguiente manera: aquellos números de la lista dada que son mayores o iguales al **umbral**, permanecen igual, pero aquellos que son menores, son reemplazados por el valor **default**.
    Por ejemplo, **lista_HomologadaPorDebajoDe_A_([3,7,8,5,1,3,2,4], 4, 2)** describe **[2,7,8,5,2,2,2,4]**.

34. Construir la función **sinDuplicados_**, que dada una lista, describa una lista que tenga todos los elementos de la lista dada, pero donde no aparecen elementos repetidos, pues las repeticiones que aparecen luego de la primera fueron eliminadas.
    Por ejemplo, **sinDuplicados_([1,3,4,2,4,3,5])** describe a la lista **[1,3,4,2,5]**. Observar que no es lo mismo describir a la lista **[1,2,4,3,5]**, que podría ser un resultado válido, pero no es el solicitado (porque no se conservó el primero de cada uno).

35. Construir las funciones

    a. **intersecciónDe_Con_**, que dadas dos listas que no contienen elementos repetidos, describe la lista de todos los elementos que aparecen en ambas.
       Por ejemplo: **intersecciónDe_Con_([1,3,4], [2,4,3,5])** describe **[3,4]**.

    b. **uniónDe_Con_**, que dadas dos listas que no contienen elementos repetidos, describe una lista sin repetidos que contenga todos los elementos que aparecen en alguna de las 2 listas.
       Por ejemplo, **uniónDe_Con_([1,3,4],[2,4,3,5])** describe **[1,3,4,2,5]**. ¿De qué tipo es la primera lista? ¿Y la segunda? ¿Pueden ser de tipos diferentes?

36. Construir la función **lista_estáIncluidaEn_** que dadas 2 listas que no contienen elementos repetidos, describe si la primer lista se encuentra contenida en la segunda lista (o sea, todos los elementos de la primera aparecen en la segunda).
    Por ejemplo, las expresiones **lista_estáIncluidaEn_([4,5],[5,3,4,6])** y **lista_estáIncluidaEn_([4,5],[2,3,4,6,5])** indican que es verdadero que está incluida, mientras que **lista_estáIncluidaEn_([4,5,8],[4,3,5,6])** indica que es falso que esté incluida.

37. Construir la función **estáOrdenada_** que, dada una lista de Números indica si está ordenada de menor a mayor. Para que una lista esté ordenada, cada elemento debe ser menor o igual al que le sigue.
    Por ejemplo, la expresión **estáOrdenada_([2,7,9,15])** indica verdadero, mientras que la expresión **estáOrdenada_([2,15,9,7])** indica falso.

    ¿La solución propuesta requirió el uso de variables que recuerdan valores diferentes a la lista que todavía falta recorrer? Si fue así, considerar ofrecer una solución que NO utilice variables además de la que se utiliza para recordar la lista a recorrer.

38. Construir la función **sinLaPrimeraAparicióndeDe_en_** que dado un *elemento* y una lista, describe la lista que se obtiene de eliminar una única vez el *elemento*, si es que este aparece en la lista.
    Por ejemplo, **sinLaPrimeraAparicióndeDe_en_(8,[4,8,42,15,8,42])** describe **[4,42,15,8,42]** y **sinLaPrimerAparicióndeDe_en_(42,[4,8,42,15,8,42])** describe **[4,8,15,8,42]**.

39. Construir las siguientes funciones:

a. **mínimoElementoDe_**, que dada una lista de Números, describe el elemento más chico que se encuentra en la lista. ¿Cuál es la precondición de la función?
Por ejemplo, **mínimoElementoDe_([13,21,3,9,45,3,7])** describe **3**.

b. **sinElMínimoElemento_**, que dada una lista de Números, describe la lista que se obtiene de eliminar una única vez el elemento más chico. ¿Cuál es la precondición de la función?
Por ejemplo, **sinElMínimoElemento_([13,21,3,9,45,3,7])** describe **[13,21,9,45,3,7]**.

c. **lista_ordenada**, que dada una lista de Números, describe la lista con los mismos elementos que la dada, pero ordenada de menor a mayor. ¿Se puede elaborar una estrategia combinando los ejercicios anteriores?
Por ejemplo, **lista_ordenada([13,21,3,9,45,17,8,3,7])** describe **[3,3,7,8,9,13,17,21,45]**.

d. ¿Puede hacerse la función anterior con alguna otra estrategia diferente a esa, que no sea simplemente utilizar la variante de funciones del ejercicio siguiente…? Este tema es un tema amplio que se tratará en detalle en la materia Estructuras de Datos.

40. Construir las siguientes funciones. ¿Cuáles son las precondiciones de estas funciones?

    a. **máximoElementoDe_**, que dada una lista de Números, describe el elemento más grande que se encuentra en la lista.
    Por ejemplo, **máximoElementoDe_([13,21,3,9,45,3,7])** describe **45**.

    b. **sinElMáximoElemento_**, que dada una lista de Números, describe la lista que se obtiene de eliminar una única vez el elemento más grande.
    Por ejemplo, **sinElMáximoElemento_([13,21,3,9,45,3,7])** describe **[13,21,3,9,3,7].**

41. Construir la función **segmentoInicialDeLargo_de_** que, dado un número que representa a una cantidad y una lista de cualquier tipo, describe la lista que se obtiene de quedarse únicamente con esa cantidad de elementos de la lista comenzando por el primero (sus primeros elementos), o todos, si no hay suficientes. ¿Cuál es la precondición de esta función? ¿Qué nombre te parece adecuado en este caso para el parámetro de tipo número? ¿Para qué tipos de lista funciona tu solución?
    Por ejemplo, **segmentoInicialDeLargo_de_(4,[4,8,15,16,99,42])** describe **[4,8,15,16]**, mientras que **segmentoInicialDeLargo_de_(2,[4,8,15,42])** describe **[4,8]** y **segmentoInicialDeLargo_de_(7,[4,8,15,42])** describe **[4,8,15,42]**.

42. Construir la función **lista_AgrupadosDeA_**, que dadas una lista de elementos cualesquiera y un número que representa a una cantidad, describe la lista de listas que agrupa los elementos de la lista dada en grupos, donde cada grupo tiene exactamente la cantidad de elementos indicada, excepto el último grupo, que puede tener menos elementos.
    Por ejemplo, **lista_AgrupadosDeA_([1,2,3,4,5,6,7,8,9,10,11],3)** describe a la lista **[[1,2,3],[4,5,6],[7,8,9],[10,11]]**. Observar que la la lista original tiene 11 elementos, mientras que la lista resultante tiene 4.

    Para realizarla, considerar utilizar la función del ejercicio anterior al pensar una estrategia. ¿Qué otra subtarea haría falta para completar dicha estrategia?

43. Construir la función **lista_Desagrupada**, que dada una lista de listas, describe una lista que contiene cada uno de los elementos de las listas internas de la lista dada.
    Por ejemplo, **lista_Desagrupada([[1,2,3],[4,5,6],[7,8,9]])** describe la lista **[1,2,3,4,5,6,7,8,9]**. Observar que la lista argumento tiene 3 elementos, mientras que la lista resultado tiene 9.

44. Construir la función **laLista_SinLosPrimeros_**, que dados una lista de elementos de cualquier tipo y un número que representa a una cantidad, describe la lista que se obtiene de quedarse con todos los elementos excepto el segmento inicial del largo dado.
    Por ejemplo, **laLista_SinLosPrimeros_([4,8,15,23,42],4)** describe **[42]**, **laLista_SinLosPrimeros_([4,8,16,23,9,42],4)** describe **[9,42]**, y la expresión **laLista_SinLosPrimeros_([4,8,15,16,23,42],8)** describe **[]**.

    ATENCIÓN: según qué estrategia se haya utilizado al escribir la función **lista_AgrupadosDeA_**, es posible que ya se haya resuelto este ejercicio como subtarea de aquél (quizás con otro nombre). En ese caso, ¿sería necesario volverlo a escribir? ¿Puede reutilizarse lo hecho con anterioridad?

45. Construir la función **sinInternos_**, que dada una lista de Números, describa la lista que se obtiene de quitar todos los elementos internos. Un elemento de una lista se dice interno si es igual al anterior de la lista.
    Por ejemplo, **sinInternos_([1,1,2,2,2,3,1,2,2])** describe **[1,2,3,1,2]**.