UNIVERSIDAD NACIONAL DE HURLINGHAM

Programación Estructurada - 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
- Algunos de 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. También 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 podrá ser utilizado en el parcial sin definirlo. Es útil mantener registro de dichos procedimientos en su carpeta.**

PRIMERAS FUNCIONES SOBRE LISTAS:

1. Una ruta por tramos

   Escribir las siguientes funciones que toman siempre un parámetro **unaRuta** que es una lista de Direcciones. Tener en cuenta que el siguiente tramo de la ruta solamente se compone de las primeras dos direcciones de la ruta si existen.

   a. **`haySiguienteTramoEn_`**, función total que indique si la ruta tiene al menos un tramo completo.

   b. **`sigueRectaEn_`**, que indique si el tramo que sigue en la ruta es una recta, asumiendo que sigue un tramo.

   c. **`sigueCurvaADerechaEn_`**, que, sabiendo que hay un tramo en la ruta, indique si lo que sigue en la ruta es una curva hacia la derecha.

   d. **`sigueCurvaAIzquierdaEn_`**, que, sabiendo que hay un tramo en la ruta, indique si lo que sigue en la ruta es una curva hacia la izquierda.

   e. **`sigueUnaCurvaEn_`**, función total que indique si el tramo que sigue en la ruta es una curva.

2. Jugando con una mano de cartas

   Escribir las siguientes funciones que reciben listas de elementos del tipo **Carta** definido en la práctica anterior. Tener en cuenta que la mano de un jugador está representada por una **Lista de Cartas** dispuestas en el orden en el que las va jugar, y el mazo es también una **Lista de Cartas**, dadas en el orden en el que se roban. No olvidarse de establecer las precondiciones necesarias en los contratos.

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

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

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

   i. **`laMano_LuegoDeRobarUnaCartaDe_`**, que dada una mano y un mazo de Cartas del cual puede robar, describir la mano resultante luego de robar.

   j. **`laMano_LuegoDeJugarUnaCarta`**, que dada una mano, describir la mano resultante luego de jugar una carta en su turno. Sí había 3 cartas en la mano, el resultado debe entonces ser una mano con 2 cartas.

   k. **`laMano_LuegoDeJugarLaSegundaCarta`**, que dada una mano, describir la mano resultante luego de jugar exclusivamente la segunda de todas sus cartas. Sí había 3 cartas en la mano, el resultado debe entonces ser una mano con 2 cartas.

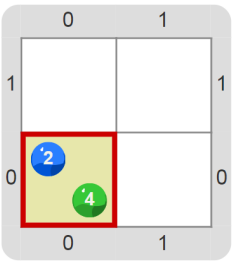
CREACIÓN DE LISTAS LEYENDO EL TABLERO:

3. Las rojas de la fila

   Escriba la función **`listaDeCantidadesDeRojasDeLaFila`** que describe una lista de números en donde cada elemento se corresponde con la cantidad de bolitas de color Rojo de una de las celdas de la fila, para cada celda de la fila de Oeste a Este.

4. Los colores en la celda actual

5. Escriba la función **`coloresEnCeldaActual()`** que describe la lista de colores de los cuales hay al menos una bolita en la celda actual. En el tablero siguiente debería describir la lista **`[Azul, Verde]`**

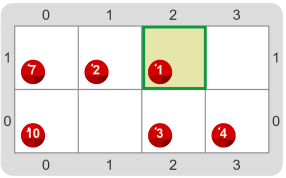


6. Los colores encontrados

   Escriba la función **`listaDeColoresEnCeldasDelTablero`** que, asumiendo que en cada celda del tablero hay bolitas de un solo color, describe una lista de colores en donde cada elemento se corresponde con el color del cual hay bolitas en esa celda, en un recorrido por celdas del tablero hacia el Sur-Oeste.

7. Apariciones de colores

   Escriba la función **`aparicionesDeColor_`** que dado un color, colorBuscado, describe una lista de números que indican para cada una de las celdas del tablero recorridas en dirección principal Norte y dirección secundaria Oeste, la cantidad de bolitas del color buscado en esa celda del tablero. Por ejemplo, para el siguiente tablero, **`aparicionesDeColor_(Rojo)`** debería describir la lista **`[4,0,3,1,0,2,10,7]`**:



8. Los colores encontrados

   Escriba la función **`direccionesAlBorde`** que retorne la lista de direcciones en las que el cabezal no se puede mover. La lista **`[Sur, Oeste]`**, deja implícito que el cabezal en este caso puede moverse al este y al norte.

FUNCIONES GENÉRICAS SOBRE LISTAS:

9. Mis primeras funciones genéricas sobre listas

Escribiremos algunas funciones genéricas que nos harán más fácil trabajar con listas más adelante. Programe entonces las siguientes funciones:

a. Escribir la función **`singular_`** que dado un elemento de cualquier tipo, describe la lista que tiene a ese único elemento.

b. Escribir la función **`esSingular_`**, que dada una Lista de elementos de cualquier tipo, indique si la lista tiene exactamente un único elemento.
SUGERENCIA: pensar en combinación con qué otras operaciones puede aprovecharse la expresión primitiva **`esVacía`**.

c. Escribir la función **`conElemento_AlFinalDe_`** que dado un elemento de cualquier tipo, y una lista de elementos de ese tipo, describa la lista que resulta de agregar el elemento dado a la lista dada como último elemento de la lista.

d. Escribir la función **`conElemento_AlComienzoDe_`** que dado un elemento de cualquier tipo, y una lista de elementos de ese tipo, describa la lista que resulta de agregar el elemento dado a la lista dada como primer elemento de la lista.

e. Escribir la función **`listaConElemento_Repetido_Veces`** que dado un elemento de cualquier tipo y un número describe una lista que tenga tantos elementos como el número dado, donde el elemento es siempre el dado.

f. Escribir la función **`listaDel_Al_`** que dados dos elementos de algún tipo enumerativo (Color, Dirección o Número) y donde se asegura que el primero es menor al segundo, describa una lista que tiene todos los elementos desde el primero al segundo, incluyendo estos. Ej. si se dan los números 1 y 5 se debe describir la lista [1, 2, 3, 4, 5], y si se dan las direcciones Norte y Oeste se debe describir la lista [Norte, Este, Sur, Oeste].

g. Escribir la función **`todosLosParesHasta_(unNumero)`** que describe la lista de números pares desde el 2 hasta el número unNumero. Ej. **`todosLosParesHasta_(20)`** describe la lista **`[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]`**

REGISTROS CON LISTAS:

10. Boletos de colectivo

Se desea modelar boletos de colectivo. Todo boleto tiene que contener ciertos datos que son obligatorios por las regulaciones vigentes. En particular, un boleto debe contener:

- El número de línea al cual pertenece el boleto (p.ej. 324, 159, 257, 85, etc.),
- La cantidad de tramos para la cual se sacó dicho boleto (siendo 1 el mínimo y 4 el máximo).
- La serie, que puede estar compuesta de números y letras (Ej. A1, 7, B).
- El número del boleto, que corresponde a un número siempre de cinco dígitos.

Con esto se pide que resuelva lo siguientes puntos:

a. Definir el tipo **`BoletoDeColectivo`**, con los datos necesarios para modelar el boleto de un colectivo según fue explicado. No olvidar escribir los tipos de los campos, y las invariantes de representación.

b. Pensar una estrategia para realizar la función **`esCapicúa_`**, que dado un boleto, determine si el número del mismo es capicúa. Un boleto es capicúa cuando el primer dígito es igual al último, y

el segundo al anteúltimo (Ej. 43834, 92729, etc.)
Reflexionamos ¿Fue fácil pensar la estrategia? ¿Con qué subtareas es conveniente contar para poder solucionar este problema fácilmente? ¿Puede usarse otra representación de **BoletoDeColectivo** que haga más fácil resolver el problema sí se brinda la siguiente función como ya realizada:

```
function reversoDe_(unaLista)
  /*
    PROPÓSITO: Describe el reverso de la lista dada, es decir, la lista dada vuelta.
      Por ej. reversoDe_([1,2,3]) es [3,2,1] y reversoDe_([Este, Norte]) es [Norte, Este].
    PARÁMETROS:
      * unaLista : Lista de A - Describe la lista dada vuelta
    PRECONDICIONES: Ninguna
    TIPO: Lista de A
    OBSERVACIÓN: "A" refiere a un tipo cualquiera, lo importante a entender es que sí
       se da una lista de una tipo, se describe una lista de ese mismo tipo.
  */
```

Asumiendo entonces la existencia de **`reversoDe_`**, realice las funciones siguientes:

c. Escribir **`costoDeBoleto_`**, que determina el costo del boleto. El costo de un boleto es de $18, si se abona un solo tramo, y aumenta en $3 por cada tramo adicional.

d. Escribir **`esCapicúaDePrimeraSerie_`**, que dado un boleto, indique si el mismo es capicúa de una primera serie. La primera serie debe ser la serie "1" o serie "A".

e. Escribir **`elBoleto_TieneElMismoNúmeroQue_`**, que dados dos boletos, **`unBoleto`** y **`otroBoleto`**, indique si ambos boletos tienen el mismo número de boleto.

f. Escribir **`elBoleto_EsParejaCapicúaDe_`**, que dados dos boletos, **`unBoleto`** y **`otroBoleto`**, indique si el número del primer boleto es igual al número del segundo boleto dado vuelta. Por ejemplo, los boletos con números 12345 y 54321 serían pareja capicúa.

g. Piense una estrategia para **`siguienteBoletoDe_`**, que dado un boleto describa otro que tenga exactamente los mismos datos que el dado (misma línea, cantidad de tramos y serie) pero donde el número es el número que sigue.
Reflexionamos ¿Es fácil de realizar una solución si se modela el número del boleto como una lista? ¿Y si lo representamos como un número? ¿Es fácil de solucionar **esCapicua_** en este escenario? Reflexione sobre cómo las operaciones que queremos poder implementar condicionan o influyen en los tipos que elegimos para modelar nuestros datos.

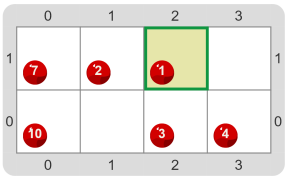
RECORRIENDO LISTAS:

11. Poniendo los colores que me piden:
Construir el procedimiento **`PonerColores_EnLaCeldaActual`**, que dada una lista de Colores, pone una bolita del color correspondiente por cada uno de los elementos de la misma. Por ejemplo, **`PonerColores_EnLaCeldaActual([Verde, Verde, Azul])`** pone dos bolitas Verdes y una Azul en la celda actual.

12. Recorriendo un camino
Construir el procedimiento **`RecorrerCamino_`**, que dada una lista de Direcciones mueve el cabezal en la dirección indicada por cada elemento, en orden. Por ejemplo, **`RecorrerCamino_([Este, Este, Norte, Este])`** debe mover el cabezal primero hacia el Este dos veces, luego hacia el Norte 1 vez y por último se mueve hacia el Este. ¿Cuál es la precondición de este procedimiento?

13. Poniendo cantidades en el tablero

Construir el procedimiento **`Poner_Bolitas_EnElTablero`**, que dada una lista de números y un color, recorre el tablero con dirección principal Norte y dirección secundaria Oeste y pone en cada celda la cantidad de bolitas del color dado, según el elemento en la posición correspondiente. Si hubiera menos números que celdas, en las celdas restantes no se pondrán bolitas, y si hubiera más números que celdas, se ignoran los números sobrantes.
Por ejemplo **Poner_Bolitas_EnElTablero([4,0,3,1,0,2,10,7],Rojo)**, si se ejecuta en un tablero inicial vacío de 4 columnas y 2 filas, produce el siguiente tablero:



FUNCIONES GENÉRICAS SOBRE LISTAS:

14. Dando vuelta la lista

BIBLIOTECA Construir la función **`reversoDe_`**, que dada una lista, describa la lista que tiene los mismos elementos que la dada, pero en orden reverso. Por ejemplo, **`reversoDe_([Negro,Azul,Azul,Verde,Rojo])`** describe **`[Rojo,Verde,Azul,Azul,Negro]`**.

15. Dando vuelta la lista

Construir las funciones

a. **BIBLIOTECA** **`longitudDe_`**, que dada una lista cualquiera, describa la cantidad de elementos de la misma.
Por ejemplo, **`longitudDe_([Azul,Azul,Verde,Rojo])`** describe **4**.

b. **BIBLIOTECA** **`sumatoriaDe_`**, que dada una lista de Números, describa la suma de todos los elementos de la misma.
Por ejemplo, **`sumatoriaDe_([1,10,15,7,9])`** describe el número **42**, porque **1+10+15+7+9** es igual a **42**;

c. **BIBLIOTECA** **`productoriaDe_`**, que dada una lista de Números, describa el producto de todos los elementos de la misma.
Por ejemplo, **`productoriaDe_([1,5,7,9])`** describe el número **315**, porque **1*5*7*9** es igual a **315**.

Reflexionamos: ¿Qué tienen en común las funciones anteriores?

16. Las transformaciones

Construir las funciones

d. **BIBLIOTECA** **`direccionesOpuestasDe_`**, que dada una lista de direcciones, describa la lista de direcciones en donde cada elemento es el opuesto al de la posición original.
Por ejemplo, **`direccionesOpuestasDe_([Oeste,Sur,Norte])`** describe **`[Este,Norte,Sur]`**.

e. **BIBLIOTECA** **`siguientesDe_`**, que dada una lista de colores, describa la lista de colores en donde cada elemento es el siguiente del original.
Por ejemplo, **`siguientesDe_([Rojo,Azul,Verde,Azul,Azul])`** describe **`[Verde,Negro,Azul,Negro,Negro]`**.
Reflexionamos: ¿De qué tipo puede ser la lista dada?

f. **BIBLIOTECA** **`elementosDe_multiplicadosPor_`**, que dada una lista de Números y un Número, que actúa de multiplicador, describe una lista donde cada número de la lista original fue multiplicado por el multiplicador.
Ej. **`elementosDe_multiplicadosPor_([1,2,3,4,5], 3)`** describe **`[3,6,9,12,15]`**.

Reflexionamos: ¿Qué tienen en común las funciones anteriores?

17. Los filtros

Construir las funciones

g. **BIBLIOTECA** **`númerosParesDe_`**, que dada una lista de Números, describa la lista de números pares que aparecen en la misma.
Por ejemplo, **`númerosParesDe_([3,4,5,2,5])`** describe **`[4,2]`**.

h. **BIBLIOTECA** **`laLista_SinElElemento_`**, que dados una lista y un elemento, describa la lista que resulta de quitar todas las apariciones del elemento dado que ocurren en lista dada. ¿De qué tipo debe ser elemento?
Por ejemplo, **`laLista_SinElElemento_([Azul,Verde,Azul,Rojo], Azul)`** describe **`[Verde,Rojo]`**.

i. **BIBLIOTECA** **`losMayoresA_De_`**, que dados un elemento "**`umbral`**" y una lista de ese elemento, describa la lista de elementos que están en la lista original que superan el umbral.
Por ejemplo, **`losMayoresA_De_(7, [3,9,7,8,5,10])`** describe **`[9,8,10]`**.

Reflexionamos: ¿Qué tienen en común las funciones anteriores?

18. Las búsquedas

Construir las funciones

j. **BIBLIOTECA** **`contiene_A_`**, que dada una lista *y* un elemento, indica si el elemento está en la lista.
Por ejemplo, **`contiene_A_([21,3,42], 3)`** indica que es verdadero, pues la lista dada contiene a **3**, mientras que **`contiene_A_([21,3,42], 5)`** indica que es falso, puesto que la lista dada no contiene a **5**.

k. **BIBLIOTECA** **`algunoMayorQué_En_`**, que dado un elemento, y una lista de elementos indica si en la lista hay algún elemento que sea mayor al dado.
Por ejemplo, **`algunoMayorQué_En_(Rojo, [Azul,Verde,Rojo,Azul])`** indica que es verdadero, pues la lista dada contiene a **`Verde`**, el cual es mayor que Rojo, mientras que **`algunoMayorQué_En_(7, [1,3,7,6,4,5])`** indica que es falso, puesto que la lista dada no contiene a ningún elemento que sea mayor a **7**.

l. **BIBLIOTECA** **`hayAlgunoDe_Entre_Y_`**, que, dada una lista de Números *y* dos números **desde** *y* **hasta**, indica si la lista contiene algún elemento que se encuentre entre los números desde y hasta, sin incluirlos. Es decir, si algún elemento *k* de la lista, cumple **desde** < *k* < **hasta**.
Por ejemplo, **`hayAlgunoDe_Entre_Y_([7,3,1,25,16], 13, 18)`** indica verdadero, pues hay un elemento entre **13** y **18** (el **16**), mientras que **`hayAlgunoDe_Entre_Y_([7,3,1,25,16], 13, 16)`** indica que es falso que haya algún elemento entre **13** y **16** (el **16** no es menor que **16**).

m. **BIBLIOTECA** **`hayAlgúnElementoImparDe_`**, que dada una lista de Números, indica si hay algún elemento de la lista que sea impar. Por ejemplo, **`hayAlgúnElementoImparDe_([2,4,5,8])`** indica verdadero, pues el 5 es impar, mientras que **`hayAlgúnElementoImparDe_([2,4,6,8])`** indica falso, pues no hay ningún número impar en la lista dada.

Reflexionamos: ¿Qué tienen en común las funciones anteriores?

19. La lista sin repetidos

BIBLIOTECA 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).

20. El índice

BIBLIOTECA Construir la función **`posiciónDe_enLaQueAparece_`** que dada una lista y un elemento, describe la posición de la lista en la que aparece ese elemento por primera vez. Se define la posición de un elemento como un número que representa la cantidad de veces que debe usarse la función **`sinElPrimero`** para acceder a ese elemento. ¿Cuál es la precondición de la función? ¿Por qué no tendría sentido que esta función sea total?
Por ejemplo, **`posiciónDe_enLaQueAparece_([4,8,15,16,42],8)`** describe al número **1**, porque solo se usa una vez **`sinElPrimero`** (para quitar el **4**), **`posiciónDe_enLaQueAparece_([4,8,15,16,42],16)`** describe **3**, porque se deben usar 3 veces la función (para quitar el 4, el 8 y el 15), y **`posiciónDe_enLaQueAparece_([4,8,15,16,23,42],4)`** describe **0**, porque no hace falta usar **`sinElPrimero`**.

21. Listas dentro de otras

BIBLIOTECA 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.

22. Listas como conjuntos

Construir las funciones

BIBLIOTECA **`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]`**.

BIBLIOTECA **`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?

23. Ordenando una lista

Construir las siguientes funciones:

BIBLIOTECA **`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`**.

BIBLIOTECA **`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]`**.

BIBLIOTECA **`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]`**.

Reflexionamos: ¿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.

24. Los máximos y sacándolos

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

BIBLIOTECA **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**.

BIBLIOTECA **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].**

APLICANDO ESQUEMAS DE LISTAS:

25. Transformando cartas

`losNumerosDeLasCartas_`, que dada una lista de Cartas, describe una lista de Números, donde cada número se corresponde al número de la carta en la lista original.
Ej. **`losNumerosDeLasCartas_(anchoDeEspada(), laCarta_de_(7,Oros), laCarta_de_(12,Copas))`** describe **`[1,7,12]`**.

26. Filtrando cartas

`soloLasDeOrosEn_`, que dada una lista de Cartas, describe una lista de Cartas que contiene solo aquellas cuyo palo es Oros.
Ej. **`soloLasDeOrosEn_(anchoDeEspada(), laCarta_de_(7,Oros), laCarta_de_(12,Copas))`** describe **`[laCarta_de_(7,Oros)]`**.

27. Búscando cartas

`hayAlgunaDeCopaEn_`, que dada una lista de Cartas, indica si hay alguna cuyo palo sea Copas.
Ej. **`hayAlgunaDeCopaEn_(anchoDeEspada(), laCarta_de_(7,Oros), laCarta_de_(12,Copas))`** indica verdadero, pues **`laCarta_de_(12,Copas)`** es de Copas, mientras que **`hayAlgunaDeCopaEn_(anchoDeEspada(), laCarta_de_(7,Oros), laCarta_de_(11,Bastos))`** indica falso, pues no hay ninguna carta de Copas en la lista dada.