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 de número 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.

6. 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:

7. 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]`.

8. 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]`.

9. Vacía o con cosas

BIBLIOTECA Escribir la función **`singular_Si_`** que dado un elemento de cualquier tipo, y un booleano que actúa de condición, describe una lista del tipo del elemento dado con las siguientes características. Si el booleano dado es verdadero, entonces la lista será una lista con un único elemento, el dado. Sí en cambio es falso, será una lista vacía.

RECORRIENDO LISTAS Y ACUMULANDO:

10. ¿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.

11. ¿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.

12. ¿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.

13. 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 que son figuras otorgan todas 20 puntos.

RECORRIENDO LISTAS Y CONSTRUYENDO OTRAS:

14. 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 `[4,8,10,12]`.

[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].

15. 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: `opuestasDe_([Norte, Este, Este])` describe la lista `[Sur, Oeste, Oeste]`.

16. 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]`

17. 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.

18. 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:

19. 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]`.

20. 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]`.

21. 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.

---

[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.

Ejemplo: `elementoDe_SinAparicionesDe_([1, 4, 1, 3, 1, 8, 9], 1)` describe la lista `[4, 3, 8, 9]`.

22. Sacando elementos

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

BUSCANDO EN LISTAS:

23. ¿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.

24. 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 `indiceEn_De_([1, 4, 2, 3, 7, 3, 9], 4)` describe 2, ya que 4 está en la segunda posición de la lista.

25. 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.

26. 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:

27. El más chico de la lista

BIBLIOTECA Escribir la función **`minimoElementoEn_`**, 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.

28. El más grande de la lista

BIBLIOTECA Escribir la función **`máximoElementoEn_`**, 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.

29. 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:

30. Todos hasta

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

31. Todos desde

BIBLIOTECA Escribir las funciones **`elementosEn_Desde_`** y **`elementosEn_DespuésDe_`**, que dada una lista de elementos y un índice (posición de la lista), describe la lista de todos los elementos en la original desde dicho índice (incluyéndolo en la primera y sin incluirlo en la segunda).
Ejemplo: `elementosEn_Desde_([5, 4, 2, 3, 7, 8, 9], 4)` describe la lista `[3, 7, 8, 9]`, es decir, todos los elementos desde la posición 4. `elementosEn_DespuésDe_([5, 4, 2, 3, 7, 8, 9], 4)` describe la lista `[7, 8, 9]`, es decir, todos los elementos que están después de la posición 4.

32. Sin elemento

BIBLIOTECA Escribir la función **`elementosEn_SinPrimeraAparicióndeDe_`**, 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_SinPrimeraAparicióndeDe_([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.
Ayuda: Puede implementarse mediante la utilización de las funciones previamente realizada.

33. 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.

34. Clonando

a. Escribir la función **`laLista_Clonada_Veces`**, que dada una lista de elementos, y un número, describe la lista que resulta de clonar tantas veces como el número dado, con todos los elementos en el orden original.

Ejemplo: `laLista_Clonada_Veces([Rojo, Verde, Azul], 3)` describe la lista `[Rojo, Verde, Azul, Rojo, Verde, Azul, Rojo, Verde, Azul]`.

b. Escribir la función **`losElementosDe_Clonados_Veces`**, que dada una lista de elementos, y un número, describe la lista que resulta de clonar tantas veces como el número dado cada uno de los elementos de la lista en el orden original.
Ejemplo: `losElementosDe_Clonada_Veces([Rojo, Verde, Azul], 3)` describe la lista `[Rojo, Rojo, Rojo, Verde, Verde, Verde, Azul, Azul, Azul]`

35. ¿Ordenada?

BIBLIOTECA 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.
Ejemplo: `estáOrdenada_([1, 7, 9, 15])` describe `Verdadero`, mientras que `estáOrdenada_([2, 15, 9, 7])` describe `Falso`.

Ayuda: ¿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.

36. Ordenando

BIBLIOTECA Escribir la función **`laLista_Ordenada`**, que dada una lista de números, describe la misma lista con los elementos ordenados.
Ejemplo: `laLista_Ordenada([2, 15, 9, 7])` describe la lista `[2, 7, 9, 15]`.

Ayuda: Se sugiere la siguiente estrategia:

a. Busque el elemento más chico en la lista y agrégalo al resultado
b. Borre el elemento más chico de la lista original.
c. Repita a y b hasta que la lista original esté vacía.

37. Algo de listas de listas

Podemos también trabajar con listas dentro de listas. Los dos primeros nos pueden ayudar en algunas soluciones que impliquen esto, y los siguientes trabajan listas de listas:

a. Escriba la función **`losPrimeros_De_`** que dado un número y una lista de elementos, describe la lista que contiene los primeros N elementos de la lista dada, donde N es el número dado.
Ejemplo: `losPrimeros_De_(3, [2, 15, 9, 7, 21])` describe la lista `[2, 15, 9]`.

b. Escriba la función **`lista_SinLosPrimeros_`** que dada una lista de elementos y un número, describe la lista original sin los primeros N elementos, donde N es el número dado.
Ejemplo: `lista_SinLosPrimeros_([2, 15, 9, 7, 21], 3)` describe la lista `[7, 21]`.

c. Escriba la función **`lista_AgrupadaDeA_`** que dada una lista de elementos y un número, describe una lista de listas en donde cada lista interna es un grupo de los siguientes N elementos de la lista original, donde N es el número dado. La última lista contiene los elementos restantes, incluso sí no forman un grupo de N elementos.
Ejemplo: `lista_AgrupadaDeA_([2, 15, 9, 7, 21, 3, 10], 3)` describe la lista de listas `[[2, 15, 9], [7, 21, 3], [10]]`.

d. Escriba la función **`lista_Desagrupada`** que dada una lista de listas de elementos, donde cada lista interna es un grupo, describe la lista que contiene a todos los elementos de cada uno de los grupos, en el orden dado por el grupo, y su posición dentro del grupo.

Ejemplo: `lista_Desagrpada([[2, 15, 9], [7, 21, 3], [10]])` describe la lista `[2, 15, 9, 7, 21, 3, 10]`.

e. Escriba la función **`combinarCadaElementoDe_Con_`** que dadas dos lista de elementos del mismo tipo, describe la lista de listas que resulta de agrupar el primer elemento de la primera lista con el primer elemento de la segunda lista, el segundo de la primera lista con el segundo de la segunda lista, etc..

Ejemplo: `combinarCadaElementoDe_Con_([2, 15, 9], [7, 21, 3])` describe la listade listas `[[2, 7], [15, 21], [9, 3]]`.

38. Listas como conjuntos

BIBLIOTECA Con listas podemos modelar conjuntos matemáticos. Podemos para ello crear un registro que "envuelva" a la lista en cuestión, y pensar operaciones para ese registro. Usaremos el siguiente tipo de datos:

```
type Conjunto is record {
  /*
    PROPÓSITO: Modela un conjunto, en el sentido matemático.
    INVARIANTE DE REPRESENTACIÓN:
      * La lista de elementos no contiene duplicados.
  */
  field elementos // Lista de "Elemento"
}
```

a. Escriba la función **`conjuntoCon_`** que dada una lista con los elementos que pertenecen al conjunto, describe un conjunto con los elementos dados. Tener en cuenta la invariante de representación de los conjuntos, que no pueden tener dos veces el mismo elemento, incluso sí este elemento aparece múltiples veces en la lista dada al momento de creación.

b. Escriba la función **`unionDe_Con_`** que dados dos conjuntos describa el conjunto resultante de unir todos los elementos del primero con todos los elementos del segundo.

c. Escriba la función **`intersecciónDe_Con_`** que dados dos conjuntos describa el conjunto resultante de intersecar todos los elementos del primero con todos los elementos del segundo, es decir, solo aquellos elementos que están en ambos conjuntos al mismo tiempo.

d. Escriba la función **`es_SubconjuntoDe_`** que dados dos conjuntos describa sí el primero de los conjuntos es un subconjunto del segundo. Es decir, sí todos los elementos del primero están en el segundo.