Algoritmos y Estructuras de Datos II

Práctica 1 – Especificación con Tipos Abstractos de Datos

Notas preliminares

- Los ejercicios marcados con el símbolo ★ constituyen un subconjunto mínimo de ejercitación. Sin embargo, aconsejamos fuertemente hacer todos los ejercicios.

Parte 1 – Axiomatización de funciones recursivas sobre TADs básicos

Ejercicio 1 *(Repaso de funciones sobre secuencias)*

Extienda el tipo Secuencia(α) definiendo las siguientes operaciones:

a) *Duplicar*, que dada una secuencia la devuelve duplicada elemento a elemento.
Por ejemplo, Duplicar(a • b • c • <>) ≡ a • a • b • b • c • c • <>.

b) • ≤ •, que chequea si una secuencia es "menor o igual" a otra según el orden lexicográfico para una relación de orden total sobre α dada[1].

c) *Reverso*, que dada una secuencia devuelve su reverso (la secuencia dada vuelta).

d) *Capicúa*, que determina si una secuencia es capicúa.
Por ejemplo, Capicúa(a • b • b • a • <>) ≡ Capicúa(1 • <>) ≡ Capicúa(<>) ≡ true.

e) *EsPrefijo?*, que chequea si una secuencia es prefijo de otra.

f) *Buscar*, que busca una secuencia dentro de otra. Si la secuencia buscada está una o más veces, la función devuelve la posición de la primera aparición; si no está, la función se indefine.

g) *EstáOrdenada?*, que verifica si una secuencia está ordenada de menor a mayor.

h) *InsertarOrdenada*, que dados una secuencia *so* (que debe estar ordenada) y un elemento e (de género α) inserta e en *so* de manera ordenada.

i) *CantidadApariciones*, que dados una secuencia y un α devuelve la cantidad de apariciones del elemento en la secuencia.

j) *EsPermutación?*, que chequea si dos secuencias dadas son permutación una de otra. Pista: utilizar *CantidadApariciones*.

k) *Combinar*, que dadas dos secuencias *ordenadas* devuelve una secuencia ordenada que resulta de juntar sus elementos. Por ejemplo, Combinar("acd", "bef") ≡ "abcdef".

Ejercicio 2 *(Funciones sobre árboles binarios)* ★

Definir las siguientes funciones sobre árboles binarios:

a) *#Hojas*, que cuenta la cantidad de hojas del árbol.

b) *DegeneradoAIzquierda*, que chequea si todo nodo interno (no hoja) tiene sólo subárbol izquierdo.

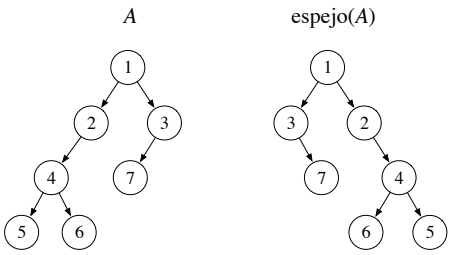
c) *ZigZag*, que chequea si todo el árbol es degenerado con direcciones alternadas.

d) *últimoNivelCompleto*, que devuelve el número del último nivel que está completo (es decir, aquél que tiene todos los nodos posibles).

[1] Por ejemplo, si representamos palabras como secuencias de letras y el orden de α (las letras) es el orden del alfabeto, $palabra_1 \leq palabra_2$ debería indicar que $palabra_1$ aparece en el diccionario antes que $palabra_2$.

e) *Espejo*, que dado un árbol devuelve su reflejo simétrico, como en un espejo. Ver figura 1(a).

f) *EsSimétrico?*, que chequea si el árbol pasado por parámetro es simétrico. Lea la nota al pie una vez que lo haya resuelto.[2]



(a) Ejemplo de *Espejo*.

Figura 1: Gráfico del ejercicio 2.

Ejercicio 3 *(Funciones sobre conjuntos)* ★

a) Definir la función recursiva *PartesDe*, que dado un conjunto de naturales devuelve un conjunto con todos los subconjuntos del conjunto parámetro.

b) Definir la función *combinacionesDeK*, que dado un conjunto C y un número natural k devuelve el conjunto de todos los subconjuntos de C que contienen k elementos.

Ejercicio 4 *(Más funciones sobre secuencias)*

Especifique la función *NTN* que dado un conjunto de secuencias y una secuencia nos devuelva el conjunto de las secuencias que:

1. son subsecuencia de la pasada como segundo parámetro, y
2. no son prefijos de ninguna otra secuencia que cumpla la propiedad anterior.

Aclaración: A **es subsecuencia de** B sii todos los elementos de A aparecen en orden no necesariamente consecutivo en B (si A tiene n elementos, entonces $A[1] = B[i_1] \dots A[n] = B[i_n]$ donde $i_1 < i_2 < \dots < i_n$).

Por ejemplo, siendo la secuencia dada 7 • 1 • 5 • 7 • 2 • 4 • 9 • <>, en la siguiente tabla se listan los resultados esperados para algunos conjuntos de secuencias posibles:

conjunto de secuencias	**resultado**
7 • 5 • 2 • <>	7 • 5 • 2 • <>
7 • 5 • 2 • <>, 7 • 5 • <>	7 • 5 • 2 • <>
7 • 5 • 2 • <>, 7 • 5 • <>, 7 • 7 • 9 • <>	7 • 5 • 2 • <>, 7 • 7 • 9 • <>
7 • 5 • 2 • <>, 7 • 5 • <>, 7 • 7 • 9 • <>, 7 • 7 • <>	7 • 5 • 2 • <>, 7 • 7 • 9 • <>
<>	<>
<>, 7 • 7 • <>	7 • 7 • <>

Ejercicio 5 *(Árboles ternarios)* ★

Un árbol ternario está definido de la siguiente forma:

[2] Si no utilizó la función *Espejo* para describir *EsSimétrico?*, vuelva a pensar el ejercicio

TAD Árbol Ternario(α)

géneros at(α)

igualdad observacional

$$(\forall a_1, a_2 : \text{at}(\alpha))\ \left(a_1 =_{\text{obs}} a_2 \iff \begin{pmatrix} \text{nil?}(a_1) =_{\text{obs}} \text{nil?}(a_2) \vee_{\text{L}} (\neg\ \text{nil?}(a_1) \Rightarrow_{\text{L}} \\ \text{raíz}(a_1) =_{\text{obs}} \text{raíz}(a_2) \wedge \text{izq}(a_1) =_{\text{obs}} \text{izq}(a_2) \\ \wedge\ \text{med}(a_1) =_{\text{obs}} \text{med}(a_2) \wedge \text{der}(a_1) =_{\text{obs}} \text{der}(a_2)) \end{pmatrix} \right)$$

generadores

nil : $\longrightarrow$ at(α)

tern : at(α) $\times$ at(α) $\times$ at(α) $\times$ α $\longrightarrow$ at(α)

observadores básicos

nil? : at(α) $\longrightarrow$ bool

raiz : at(α) a $\longrightarrow$ α {$\neg$nil?(a)}

izq : at(α) a $\longrightarrow$ at(α) {$\neg$nil?(a)}

med : at(α) a $\longrightarrow$ at(α) {$\neg$nil?(a)}

der : at(α) a $\longrightarrow$ at(α) {$\neg$nil?(a)}

Fin TAD

Escribir las siguientes funciones sobre árboles ternarios:

a) *NivelNormal?*, que dado un árbol ternario y un entero k devuelve verdadero si todos los nodos en el nivel k tienen exactamente 3 hijos no *nil*. En caso contrario, esta función devuelve falso.
b) *Acotado?*, que dado un árbol ternario y un entero k devuelve verdadero si todo nivel en el árbol tiene como máximo k nodos. *Acotado?* devuelve falso en caso contrario.

Ejercicio 6 *(Funciones sobre* rosetrees*)* ★

A continuación se presenta el tipo Rosetree:

TAD Rosetree(α)

géneros rosetree(α)

igualdad observacional

$$(\forall r_1, r_2 : \text{rosetree})\ (r_1 =_{\text{obs}} r_2 \iff (\text{raíz}(r_1) =_{\text{obs}} \text{raíz}(r_2) \wedge \text{hijos}(r_1) =_{\text{obs}} \text{hijos}(r_2)))$$

observadores básicos

raíz : rosetree(α) $\longrightarrow$ α

hijos : rosetree(α) $\longrightarrow$ secu(rosetree(α))

generadores

rose : α $\times$ secu(rosetree(α)) $\longrightarrow$ rosetree(α)

axiomas $\forall s$: secu(rosetree(α)) $\forall a$: α

raíz(rose(a, s)) $\equiv$ a

hijos(rose(a, s)) $\equiv$ s

Fin TAD

Ejemplos de instancias:
rose(1, <>)
rose(true, rose(false, <>) • <>)
rose(true, rose(false, <>) • rose(true, rose(true, <>) • <>) • rose(false, <>) • <>)

Defina operaciones auxiliares para:

a) Conocer la altura de un rosetree.
b) Calcular la cantidad de hojas de un rosetree.
c) Podar un rosetree, es decir, eliminar todas las hojas del rosetree.

d) Obtener todas las ramas de un rosetree cuya longitud sea menor o igual a un valor n dado (Una rama es una secuencia de nodos del árbol que va desde la raíz hasta una de sus hojas).
e) Dado un número natural n, devolver una secuencia con los elementos del nivel n enumerados de izquierda a derecha (Los elementos de nivel n son aquellos nodos que se encuentran a distancia n de la raíz del rosetree).
f) Calcular el conjunto de las ramas más largas de un rosetree que contienen al menos un elemento repetido .

Parte 2 – Especificación con Tipos Abstractos

Ejercicio 7 *(Polinomios)* ★

Se quieren especificar los polinomios de coeficientes naturales, sabiendo que:

- un número natural es un polinomio.
- la indeterminada X es un polinomio.
- si p_1 y p_2 son dos polinomios, entonces $p_1 + p_2$ y $p_1 \cdot p_2$ son polinomios.

Se desea tener en el tipo una operación de evaluación, que dado un polinomio y un número natural lo evalúe de la manera habitual y otra que indique si un número natural es raíz de un polinomio. Completar la especificación a partir de la signatura dada.

TAD POLINOMIO

géneros polinomio

igualdad observacional
$(\forall p_1, p_2 : \text{polinomio})\ (p_1 =_{\text{obs}} p_2 \iff ((\forall n : \text{nat})(\text{Evaluar}(p_1, n) =_{\text{obs}} \text{Evaluar}(p_2, n))))$

observadores básicos
Evaluar : polinomio × nat ⟶ nat

generadores
Cte : nat ⟶ polinomio
X : ⟶ polinomio
$\bullet + \bullet$: polinomio × polinomio ⟶ polinomio
$\bullet \cdot \bullet$: polinomio × polinomio ⟶ polinomio

axiomas ...

Fin TAD

Ejercicio 8 *(Robot)*

Especifique tipos para un robot que realiza un camino a través de un plano de coordenadas cartesianas (enteras), es decir, tiene operaciones para ubicarse en un coordenada, avanzar hacia arriba, hacia abajo, hacia la derecha y hacia la izquierda, preguntar por la posición actual, saber cuántas veces pasó por una coordenada dada y saber cuál es la coordenada más a la derecha por dónde pasó.

Completar la especificación a partir de la siguiente signatura:

TAD ROBOT

igualdad observacional
$(\forall r_1, r_2 : \text{robot})\ (r_1 =_{\text{obs}} r_2 \iff (\text{Trayectoria}(r_1) =_{\text{obs}} \text{Trayectoria}(r_2)))$

observadores básicos
Trayectoria : robot ⟶ secuencia(coordenada)

generadores
Ubicar : coordenada ⟶ robot
Arriba : robot ⟶ robot
Abajo : robot ⟶ robot

Derecha : robot ⟶ robot
Izquierda : robot ⟶ robot

otras operaciones
PosiciónActual : robot ⟶ coordenada
CuántasVecesPasó : coordenada × robot ⟶ nat
MásALaDerecha : robot ⟶ coordenada

axiomas ...

Fin TAD

TAD Coordenada es Tupla(entero × entero)

Ejercicio 9 *(Electroimán)* ★

Se dispone de una cinta circular, dividida en una cantidad fija de celdas, sobre las cuales pueden o no existir elementos metálicos. Esta cinta se mueve en ambos sentidos.

Sobre la cinta se encuentra montado un electroimán que atrae los elementos metálicos de la cinta cuando éstos quedan debajo de él. La disposición inicial de los elementos sobre la cinta puede ser cualquiera.

El imán puede estar prendido o apagado, y sólo atrae elementos cuando está prendido y no está cargado (ocupado) con algún elemento previamente atraído. Además, sólo puede apagarse si el objeto atraído (de haberlo) puede ser depositado debajo en la cinta.

Tanto el imán como cada celda de la cinta sólo pueden contener un elemento a la vez, es decir que si el imán tiene atraído un elemento no puede atraer otro, y que si sobre una celda hay un elemento no puede soltarse sobre ella el elemento que tenga atraído el imán.

Por convención numeraremos las celdas de la cinta de 0 a n-1, siendo n la cantidad total de celdas. Además suponemos que la celda inicial es la número 0.

Se desea conocer cuántas veces se aplicaron las funciones "girar el imán" a la izquierda y a la derecha (← y → respectivamente), por separado, y cuántos elementos metálicos hay sobre una cinta.

Completar la especificación a partir de la signatura dada.

TAD Electroimán

igualdad observacional
$(\cdots)$

observadores básicos
Cinta : electroimán ⟶ cinta
ImánPrendido? : electroimán ⟶ bool
ImánCargado? : electroimán e ⟶ bool {ImánPrendido?(e)}

generadores
Arrancar : cinta ⟶ electroimán
Prender : electroimán e ⟶ electroimán {¬ImánPrendido?(e)}
Apagar : electroimán e ⟶ electroimán
{ImánPrendido?(e) $\wedge_{\text{L}}$ (ImánCargado?(e) ⇒ ¬CeldaActualOcupada?(e))}
← : electroimán ⟶ electroimán
→ : electroimán ⟶ electroimán

otras operaciones
CeldaActualOcupada? : electroimán ⟶ bool
#Giros← : electroimán ⟶ nat
#Giros→ : electroimán ⟶ nat

axiomas ...

Fin TAD

TAD CINTA

igualdad observacional
$(\cdots)$

observadores básicos

#Celdas	: cinta	$\longrightarrow$	nat		
CeldaOcupada?	: nat n × cinta c	$\longrightarrow$	bool		$\{n < \#\text{Celdas}(c)\}$
CeldaActual	: cinta	$\longrightarrow$	nat		
#Giros←	: cinta	$\longrightarrow$	nat		
#Giros→	: cinta	$\longrightarrow$	nat		

generadores

Arrancar	: nat n	$\longrightarrow$	cinta	$\{n > 0\}$
PonerElem	: cinta c	$\longrightarrow$	cinta	$\{\neg\text{CeldaActualOcupada?}(c)\}$
SacarElem	: cinta c	$\longrightarrow$	cinta	$\{\text{CeldaActualOcupada?}(c)\}$
←	: cinta	$\longrightarrow$	cinta	
→	: cinta	$\longrightarrow$	cinta	

otras operaciones

CeldaActualOcupada?	: cinta	$\longrightarrow$	bool
#Elem	: cinta	$\longrightarrow$	nat

axiomas ...

Fin TAD

Ejercicio 10 *(Para hacer en la fila del banco)* ★

Con el objetivo de mejorar su servicio, un banco decidió analizar el comportamiento de sus largas filas de clientes. Se encargó la tarea al seor Felipe N. Sativo, empleado de la casa central del banco en cuestión, quien inmediatamente reconoció la necesidad de pedir nuestro asesoramiento.

Proponer una solución para cada item del ejercicio, discutirlo con sus compaeros y luego consultarlo con un docente.

a) El seor Sativo nos presentó un informe en el que detallaba el comportamiento esperado de una fila de clientes. Después de leerlo, releerlo y extraer de allí la información importante, comprendimos que las acciones que esperaba registrar eran la apertura de la ventanilla, la llegada de un nuevo cliente a la fila, y la atención del cliente que estuviera en primer lugar (con su consecuente egreso de la fila). También quedó claro que el banco deseaba poder verificar si la fila estaba vacía, conocer la longitud (cantidad de clientes) de la fila, si un cliente determinado estaba o no esperando su atención en dicha fila y, en caso de estarlo, cuál era su posición en la misma (siendo la posición 1 la del próximo cliente que sería atendido, es decir, el que haya llegado primero entre los clientes presentes).

Especificar (distinguiendo generadores, observadores básicos y otras operaciones, y escribiendo los axiomas) el tipo FILA, cuyas funciones a exportar son las siguientes:

AbrirVentanilla	:	$\longrightarrow$	fila	
Llegar	: persona p × fila f	$\longrightarrow$	fila	$\{\neg\text{Esperando}(p, f)\}$
Atender	: fila f	$\longrightarrow$	fila	$\{\neg\text{Vacía}(f)\}$
Esperando	: persona × fila	$\longrightarrow$	bool	
Vacía	: fila	$\longrightarrow$	bool	
Posición	: persona p × fila f	$\longrightarrow$	nat	$\{\text{Esperando}(p, f)\}$
Longitud	: fila	$\longrightarrow$	nat	

El género es "fila". Se permite agregar funciones auxiliares en caso de ser necesario.

b) Durante el primer día de observación, Felipe N. Sativo descubrió que la realidad de las filas del banco era más compleja de lo que él había previsto. Notó que algunos clientes, desalentados por la longitud y lentitud de la fila, se retiraban de la misma sin haber sido atendidos. También detectó algunos clientes que no respetaban el orden de la fila, introduciéndose en ella delante de otros clientes que habían llegado antes (en términos

más simples: colándose). Felipe decidió que era importante registrar estos sucesos, así como también conocer la identidad de los irrespetuosos alteradores del orden.

Modificar la especificación anterior para incluir las siguientes funciones:

Retirarse	: persona p × fila f	⟶ fila	{Esperando(p, f)}
ColarseAdelanteDe	: persona p × persona q × fila f	⟶ fila	{¬Esperando(p, f) ∧ Esperando(q, f)}
SeColó?	: persona p × fila f	⟶ bool	{Esperando(p, f)}

c) El banco felicitó al Sr. Sativo por su gran desempeo, y le entregó un último encargo, necesario para comprobar la verdadera eficiencia del banco con respecto a la atención de sus clientes. El mismo consistía en averiguar si un cliente dado ingresó alguna vez a la fila (ya se encuentre esperando en ella, haya sido atendido o se haya retirado), y si una persona determinada había sido atendida durante el día (nótese que el tipo FILA sólo registra la información de un día, desde el momento en que se abre la ventanilla).

Modificar nuevamente la especificación para agregar las funciones faltantes:

Entro?	: persona × fila	⟶ bool
FueAtendido?	: persona × fila	⟶ bool

Notar que la nueva especificación puede no cumplir los puntos anteriores.

Ejercicio 11 *(Biblioteca)*

a) En una biblioteca se guarda información sobre libros y sus autores. Se quiere poder encontrar: para un libro dado, su autor, y para un autor dado, todos los libros que escribió. Especifique este tipo.
b) Suponga que los libros se registran por medio de su ISBN (International Standard Book Number). Modifique la especificación para reflejar este cambio en el modelo, de manera que la biblioteca pueda contener libros distintos (i.e., con distinto ISBN), pero con idéntico título.

Ejercicio 12 *(Stock)*

Especifique un tipo abstracto de datos Stock que permita las siguientes funcionalidades:

- Registrar un nuevo producto con su stock mínimo (un natural).
- Registrar las compras y las ventas de un producto, indicando la cantidad de elementos comprados y vendidos, respectivamente.
- Permitir que sea informado un producto sustituto para otro dado.
- Devolver el conjunto de todos los productos con stock debajo del mínimo que no tengan sustituto, o bien tales que el total del stock del producto más el de su sustituto esté por debajo del mínimo.

Ejercicio 13 *(Videoclub)*

Especifique tipos para administrar los alquileres de un videoclub. El videoclub, bastante nuevo, no cuenta aún con sucursales. Se desea conocer el stock de los videos, cuántas copias de un video están alquiladas y no las han devuelto, y quiénes las alquilaron. Además se desea registrar la dirección de correo electrónico de los clientes.

Por simplicidad se supone que los clientes se identifican por su nombre, y que el catálogo de videos está descripto por los videos que se van comprando.

Ejercicio 14 *(Red caminera)*

Modelar y especificar el siguiente problema. Se trata de una empresa transportista que desea conocer la situación de su flota de camiones. Esta flota de camiones se desplaza a través de diversas ciudades, llevando la carga que la empresa transportista le asigne en cada viaje. El tipo red caminera ya está definido, y es el que modela las ciudades y las distancias entre ellas, que por simplicidad se toman como naturales. El tipo a especificar debe recibir una red caminera para generar una instancia del tipo y luego se podrán informar:

a) El comienzo de servicio para la empresa de un nuevo camión, y en qué ciudad comienza sus actividades.
b) Los movimientos de los camiones, que reflejan cambios en su posición, siempre respetando la red caminera definida.

Escribir los axiomas y la igualdad observacional, teniendo en cuenta que interesa el recorrido de los camiones. Se deberán además proveer operaciones para conocer la cantidad de kilómetros efectuados por un camión desde que inició sus servicios, y cuáles fueron las ciudades más visitadas.

TAD RED

igualdad observacional
$(\cdots)$

observadores básicos
Ciudad? : ciudad $\times$ red $\longrightarrow$ bool
Conectadas? : ciudad c_1 $\times$ ciudad c_2 $\times$ red r $\longrightarrow$ bool $\{\text{Ciudad?}(c_1,\text{r}) \land \text{Ciudad?}(c_2,\text{r}) \land c_1 \neq c_2\}$
Distancia : ciudad c_1 $\times$ ciudad c_2 $\times$ red r $\longrightarrow$ nat
$\{\text{Ciudad?}(c_1,r) \land \text{Ciudad?}(c_2,r) \land c_1 \neq c_2 \land_{\text{L}} \text{Conectadas?}(c_1,c_2,r)\}$

generadores
Pavimentar : $\longrightarrow$ red
Edificar : ciudad c $\times$ red r $\longrightarrow$ red $\{\neg\text{Ciudad?}(c,r)\}$
Conectar : ciudad c_1 $\times$ ciudad c_2 $\times$ nat d $\times$ red r $\longrightarrow$ red
$\{\text{Ciudad?}(c_1,r) \land \text{Ciudad?}(c_2,r) \land c_1 \neq c_2 \land\ (d > 0) \land_{\text{L}} \neg\text{Conectadas?}(c_1,c_2,r)\}$

otras operaciones
Ciudades : red $\longrightarrow$ conjunto(ciudad)
...

Fin TAD

TAD CIUDAD **extiende** NAT
con el agregado de la operación (aquí se adopta esta abreviatura para no sobrecargarlos en lecturas)

otras operaciones
Match : ciudad c_1 $\times$ ciudad c_2 $\times$ ciudad c_3 $\times$ ciudad c_4 $\longrightarrow$ bool $\{c_1 \neq c_2 \land c_3 \neq c_4\}$

axiomas $\forall$ ciudad c_1, c_2, c_3, c_4
$\text{Match}(c_1,c_2,c_3,c_4) \equiv ((c_1 = c_3) \land\ (c_2 = c_4)) \lor ((c_1 = c_4) \land (c_2 = c_3))$

Fin TAD

Ejercicio 15 *(Taller de reparaciones)* ★

Se trata de un taller que realiza reparaciones (no gratuitas) de artículos electrónicos. Las reparaciones que realiza están definidas por la gerencia del taller, y no admiten discusión alguna.

Un taller tiene un catálogo con toda la gama de reparaciones que realiza. Cada reparación se define por los repuestos que consume su realización, si es que consume alguno, más un costo de mano de obra. Se tiene también una (única) lista de precios de los repuestos que se utilizan.

Por un lado, el taller tiene un stock de repuestos que utiliza para realizar reparaciones, en donde cada tanto se reponen repuestos (esto puede ocurrir en cualquier momento). Por otra parte, recibe órdenes de trabajo, que no son otra cosa que las reparaciones a realizar. (Se debe diferenciar la orden de trabajo de su realización.)

Los tipos catálogo, reparación y precios se dan en parte definidos, y son los que modelan el catálogo de reparaciones que el taller puede realizar, una reparación y la lista de precios de los repuestos, respectivamente. **NOTA:** estos tipos pueden extenderse de ser necesario.

Especificar un tipo que complete el modelo del problema, incluyendo las siguientes funciones:

- Ejecutar una orden de trabajo, si ésta es factible.
- Conocer cuáles son las reparaciones (es decir, sus códigos) más solicitadas, y sus precios.

Por simplicidad puede suponerse que cada orden de trabajo consiste en una única reparación.

TAD CATÁLOGO

observadores básicos
Catalogado? : código × catálogo ⟶ bool
Reparación : código *cod* × catálogo *cat* ⟶ reparacion {Catalogado?(*cod*, *cat*)}

generadores
EnBlanco : ⟶ catálogo
Catalogar : código *cod* × reparacion × catálogo *cat* ⟶ catálogo {¬Catalogado?(*cod*, *cat*)}

axiomas ∀ catálogo *cat*, ∀ código cod, cod_1, ∀ reparacion *rep*
Catalogado?(*cod*, EnBlanco) ≡ false
Catalogado?(*cod*, Catalogar(cod_1, *rep*, *cat*)) ≡ $(cod = cod_1) \vee$ Catalogado?(*cod*, *cat*)

Reparación(*cod*, Catalogar(cod_1, *rep*, *cat*)) ≡ **if** $cod = cod_1$ **then** *rep* **else** Reparación(*cod*, *cat*) **fi**

Fin TAD

TAD CÓDIGO **es** STRING

TAD REPARACIÓN

observadores básicos
CostoManoObra : reparación ⟶ nat
Consume? : repuesto × reparación ⟶ bool
Cantidad : repuesto *rto* × reparación *rep* ⟶ nat {Consume?(*rto*, *rep*)}

generadores
Definir : nat *c* ⟶ reparación {$c > 0$}
Consumir : repuesto × reparación ⟶ reparación

axiomas ∀ reparación *rep*, ∀ repuesto rto, rto_1, ∀ nat *c*
CostoManoObra(Definir(*c*)) ≡ *c*
CostoManoObra(Consumir(*rto*, *rep*)) ≡ CostoManoObra(*rep*)

Consume?(*rto*, Definir(*c*)) ≡ false
Consume?(*rto*, Consumir(rto_1, *rep*)) ≡ $(rto = rto_1) \vee$ Consume?(*rto*, *rep*)

Cantidad(*rto*, Consumir(rto_1, *rep*)) ≡ [**if** $rto = rto_1$ **then** 1 **else** 0 **fi**] +
[**if** Consume?(*rto*, *rep*) **then** Cantidad(*rto*, *rep*) **else** 0 **fi**]

Fin TAD

TAD PRECIOS

observadores básicos
TienePrecio? : repuesto × precios ⟶ bool
PrecioUnitario : repuesto *rto* × precios *pr* ⟶ nat {TienePrecio?(*rto*, *pr*)}

generadores
SinPrecios : ⟶ precios
Tasar : repuesto *rto* × nat *c* × precios *pr* ⟶ precios {¬TienePrecio?(*rto*, *pr*) ∧ $c > 0$}

axiomas ∀ precios *pr*, ∀ repuesto rto, rto_1, ∀ nat *c*
TienePrecio?(*rto*, SinPrecios) ≡ false
TienePrecio?(*rto*, Tasar(rto_1, *c*, *pr*)) ≡ $rto = rto_1 \vee$ TienePrecio?(*rto*, *pr*)

PrecioUnitario(*rto*, Tasar(rto_1, *c*, *pr*)) ≡ **if** $rto = rto_1$ **then** *c* **else** PrecioUnitario(*rto*, *pr*) **fi**

Fin TAD

TAD Repuesto **es** String

Ejercicio 16 *(Juego de la oca)* ★

Especifique el juego de la oca.

Dos jugadores, *alternadamente*, se mueven por un camino de baldosas, numeradas éstas $0, 1, 2, \ldots$ avanzando en cada turno tantas baldosas como indique el dado tirado y realizando la acción que indica el tablero para la baldosa alcanzada. Van jugando de a uno por vez.

El tablero indica que ambos jugadores comienzan en la baldosa cero, el número de la baldosa en donde está la llegada (que debe ser distinto del de partida) y cómo debe moverse un jugador cuando llega a una determinada baldosa: cuántas baldosas avanzar o retroceder, o no hacer nada (en los demás casos). La acción correspondiente a una baldosa sólo se toma si se llega a esa avanzando con el dado, no si se llega mediante la acción de otra baldosa. Gana el primer jugador que alcanza la llegada. Proveer también funciones para saber qué indicó el dado en cada jugada para cada uno de los jugadores, y para saber si un jugador pasó por una baldosa determinada.

Pista: Definir al menos 2 tipos; uno para el tablero y otro para el juego de la oca propiamente dicho, que use el tablero.

Pista 2: Recuerde que los TADs no nos permiten modelar eventos aleatorios.

Ejercicio 17 *(Colectivo con asientos para personas con movilidad reducida)* ★

Un colectivo realiza un recorrido fijo (es decir, tiene una cantidad predefinida de paradas). El colectivo posee una cantidad fija de asientos, dispuestos en filas de dos. Cuando una persona se sube, se sienta en alguna de las butacas libres. Si no hubiera más asientos libres, se queda parada. Al bajarse, cosa que sucede cuando el colectivo llega a la parada indicada al subir, los asientos ocupados por los pasajeros son inmediatamente ocupados por algunas de las personas paradas. Es decir, habiendo asientos libres nunca hay pasajeros parados.

Se pide:

a) Modelar el TAD Colectivo, presentando:

- Igualdad observacional.
- Observadores básicos.
- Generadores.
- Otras operaciones.
- Axiomas.

b) Cómo cambiaría el TAD del punto anterior si se agregara el siguiente párrafo?

> Además pueden subir personas con movilidad reducida (embarazadas, discapacitados, etc). Cuando sube una persona con movilidad reducida y no hay asientos libres, alguna de las personas que esté sentada más adelante y no tenga movilidad reducida le deberá ceder el asiento. Esta persona, obviamente, continua el viaje parada.

Describa el comportamiento de todas las funciones que cambien. Puede axiomatizar si lo desea para aportar mayor claridad, pero no es un requisito.

Ejercicio 18 *(Banco con clientes prioritarios)*

Un banco posee dos cajas y una única cola. La caja A, atiende sólo gente *bien*, mientras que la caja B atiende a los *proles*, siempre y cuando no haya en la cola gente *bien*, que tiene prioridad en esa caja. Por razones de seguridad las personas deben identificarse dando su número de DNI al entrar al banco, momento en el cuál se ponen en la cola. Pueden cansarse de esperar y retirarse antes de ser atendidas. Cuando los cajeros están libres gritan "pase el que sigue", momento en el que atienden al próximo de la cola según el criterio de cada caja, que se retira sin dilaciones al terminar su transacción.

A los cajeros se les paga por cada operación realizada, de manera que interesa saber a quiénes atendió cada cajero, y por razones de seguridad, es necesario tener un control estricto de quienes entraron y salieron del banco.

Se pide:

a) Modelar el TAD Banco, presentando:

- Igualdad observacional.
- Observadores básicos.
- Generadores.
- Otras operaciones.
- Axiomas.

b) Cómo cambiaría el TAD del punto a) si se quisiera modelar una cantidad no determinada a priori de cajas de cada tipo y si las personas pudieran colarse? Describa el comportamiento de todas las funciones que cambien.

Ejercicio 19 *(Gestión de baos en un parcial)*

Con el fin de garantizar que dos alumnos no se encuentren en el bao mientras están realizando un parcial, se decidió implementar la siguiente política. Cuando un alumno solicita ir al bao, este puede concurrir al mismo si ningún otro alumno está en el bao. En caso contrario, éste esperará cerca de la puerta. Un alumno que está esperando cerca de la puerta puede decidir volver a sentarse en cualquier momento. Cuando un alumno regresa del bao, automáticamente va el primero de los alumnos que está esperando cerca de la puerta. Con el fin de no hacer perder mucho tiempo a los alumnos esperando en la cola, si cuando vuelve la persona que estaba en el bao la fila tiene 5 o más alumnos esperando, entonces todos juntos van al bao acompaados por un docente.

Por otra parte, no se permite a un alumno ir mas de 3 veces al bao durante la realización del parcial.

Modelar mediante TADs el problema descripto anteriormente, teniendo en cuenta que se desea conocer en todo momento el alumno que se encuentra en el bao.

Ejercicio 20 *(Parte del suite de juegos de su S. O. preferido)* ★

Se desea modelar el juego de mesa que se describe a continuación usando TADs. El mismo se juega sobre un tablero cuadrado de 64×64 posiciones. En este juego participan siempre 64 jugadores. El juego comienza cuando los jugadores se posicionan en el tablero: esto es, eligen una celda del tablero y una dirección (norte, sur, este, oeste). No se admiten dos jugadores posicionados en la misma celda. El puntaje inicial de cada jugador es 0.

En cada turno del juego algún jugador avanza o cambia de dirección. Si avanza, se desplaza a la celda siguiente de acuerdo con la dirección que tiene el jugador. Si se encuentra posicionado en un extremo del tablero y tiene una dirección que lo lleva fuera del tablero, al avanzar se posiciona en el otro extremo del tablero (por ejemplo, si se encuentra en la celda más al norte de la columna i y tiene dirección norte, al avanzar se desplaza a la celda más al sur de la columna i). Si al desplazarse, un jugador llega a una celda que está ocupada por otro jugador, este último es eliminado y quién se movió gana todos los puntos que había acumulado el jugador eliminado más un bonus de 1 punto. Si cambia de dirección, el jugador no se desplaza.

En cada turno, se elije no determinísticamente el jugador que tiene derecho a moverse o cambiar de dirección. El juego termina cuando un jugador alcanza un puntaje que es superior a 32.

Puede asumir que se encuentran definidos los tipos Posicion, Direccion y Tablero, con Tablero definido de la siguiente manera:

TAD Tablero(Extracto)

géneros tablero

generadores

nuevo : $\longrightarrow$ tablero

otras operaciones

siguiente : posicion p $\times$ direccion $\times$ tablero t $\longrightarrow$ posicion {posicionValida(p, t)}

posicionValida : posicion $\times$ tablero $\longrightarrow$ bool

Fin TAD

Parte 3 – Ejercicios de parcial

Dejamos a su disposición algunos ejercicios de parciales. De todas maneras recomendamos realizarlos luego de resolver los ejercicios de la guía.

Ejercicio 21 *(La red antisocial)*

Se desea modelar mediante TADs el comportamiento de la "La red antisocial". Esta red está generada por un ente, llamado líder, que se encarga de convocar a distintos "seguidores" a la red. Cada uno de estos seguidores a la vez puede convocar otros seguidores para que se unan a la red. En cualquier caso, todo miembro fue convocado por algún miembro preexistente.

En cualquier momento puede ocurrir que un miembro (incluido el líder) sea enjuiciado por la red. En dicho juicio se conforma un tribunal que dictamina si hay que echarlo o no. El tribunal está constituidos por tres miembros elegidos de alguna forma (que se decidirá luego en la etapa de diseo) entre todos los integrantes de la red.

El dictamen del tribunal se puede saber directamente cuando éste se conforma: si más de la mitad del tribunal "desciende" del enjuiciado, es decir, fue convocado por él o uno de sus convocados, entonces no se lo expulsará. De otra forma se expulsará no sólo al enjuiciado, sino también a todos los miembros a quienes el expulsado haya convocado y a los convocados de los convocados (y así recursivamente).

Una vez expulsado (ya sea directa o recursivamente), cada ex-miembro queda prohibido de volver a pertenecer a la red de por vida. Claro, de por vida es mucho tiempo, sobre todo para aquellos expulsados que no fueron enjuiciados. Entonces, la red tiene un esquema de absoluciones. Si un ex-miembro que fue expulsado es absuelto, entonces se elimina cualquier registro en la red acerca de que dicha persona fue alguna vez miembro de la red y entonces puede ser nuevamente convocado a unirse.

Interesa determinar, además, la cantidad total de seguidores que tiene cada miembro de la red.

Ejercicio 22 *(La oficina postal)*

Se quiere especificar un sistema para controlar una oficina postal que se ocupa de almacenar, administrar y entregar la correspondencia en una ciudad. La ciudad en cuestión se encuentra dividida en zonas, cada una de las cuales se identifica con un código postal. La oficina recibe periódicamente un cargamento de paquetes (cartas, encomiendas, documentos, etc.). Cada paquete tiene escrito el código postal de su destinatario y un peso en gramos.

La oficina cuenta con un grupo de empleados destinados al reparto de paquetes (los *carteros*). A cada cartero le corresponde entregar los paquetes dentro de una única zona. Es decir, cada cartero tiene asociado un único código postal.

Cuando llega un camión con un cargamento, los empleados de la oficina organizan los paquetes recibidos. Los paquetes cuyo código postal no se corresponde con el de algún cartero se devuelven inmediatamente al camión. Los paquetes restantes ingresan al depósito del correo.

Un cartero puede iniciar su recorrido en cualquier momento, llevándose del depósito una bolsa de paquetes para repartir. Los paquetes a repartir se asignan de la manera detallada a continuación, teniendo en cuenta que el método utilizado para seleccionarlos se postergará hasta el momento de la implementación:

- A cada cartero le corresponden únicamente paquetes asociados a su código postal.
- El peso total de la bolsa no debe exceder los 25Kg.

Cuando el cartero finaliza su recorrido, reporta cuáles paquetes no pudieron ser entregados. Los paquetes *rebotados* se reingresan al depósito. Este conjunto debe ser (obviamente) un subconjunto de los paquetes que le correspondía entregar al momento de iniciar su recorrido, en caso de que sea el mismo conjunto, el cartero es **despedido** de la oficina postal.

Especificar el sistema utilizando TADs. Se desea saber en todo momento:

- Cuántos paquetes hay en el depósito, y qué características tienen.
- Cuántos paquetes fueron rebotados.