



Cuestión 1

1.5 puntos

Un puzzle móvil es un juego que consiste en ordenar las piezas de un tablero, en el que siempre hay un hueco, hasta conseguir una determinada figura. Se desea diseñar una variante de este juego, con las siguientes características:

- El tablero tiene un tamaño de 5x5, es decir, podemos poner hasta 24 piezas.
- Al comenzar el juego, el tablero está vacío.
- Una vez colocada una pieza, ésta puede desplazarse en los 4 sentidos (derecha, izquierda, arriba y abajo). Para simplificar sólo se especificará el desplazamiento hacia la derecha.

Para facilitar el diseño del juego, se desea crear el TAD *PUZZLE MÓVIL*, con un conjunto de operaciones, cuya signatura es la siguiente:

espec PUZZLE MÓVIL

usa cadenas, naturales, booleanos

género puzM

operaciones

puzVacío: \rightarrow puzM

{crea un puzzle vacío}

parcial ponerP: puzM natural natural \rightarrow puzM

{coloca una pieza en la coordenada (x,y). No se puede poner una pieza en una coordenada ocupada}

numP: puzM \rightarrow natural

{devuelve el número de piezas que tiene actualmente el puzzle}

parcial ocupada: puzM natural natural \rightarrow booleano

{devuelve verdadero si la posición (x,y) está ocupada y falso en caso contrario}

parcial esMóvil: puzM natural natural \rightarrow booleano

{devuelve verdadero si la pieza situada en la posición (x,y) se puede desplazar en alguno de los cuatro sentidos y falso en caso contrario}

parcial moverD: puzM natural natural \rightarrow puzM

{desplaza a la derecha la pieza que se encuentra en la posición (x,y). Se produce un error en el caso de que la posición de la derecha esté ocupada y que la pieza que se desea mover no esté en el puzzle}

parcial quitarP: puzM natural natural \rightarrow puzM

{quita la pieza que se encuentra en la posición (x,y)}

Considerando $G = \{\text{puzVacío}, \text{ponerP}\}$ el conjunto de operaciones generadoras, completar la especificación algebraica del TAD *PUZZLE MÓVIL* con los dominios de definición de las operaciones parciales y las ecuaciones.

Cuestión 2

1 punto

Supongamos la siguiente representación dinámica para almacenar un Árbol Binario de Búsqueda.

tipo ABB = **puntero a** nodo;
nodo = **registro**
valor: entero;
izq, der: ABB;
fregistro;

Escribir, en lenguaje algorítmico y de forma recursiva, las siguientes operaciones:

función enlacesNulos (A: ABB): entero;
{devuelve el número de enlaces vacíos que tiene el árbol A}

función niveles (A: ABB): entero;
{devuelve el número de niveles que tiene el árbol A}

Cuestión 3

1.5 puntos

Se pretende resolver un determinado problema usando programación modular basada en TAD's. Para ello se dispone del TAD **Almacén**, en el que se define el tipo opaco *Caja*, y cuyo módulo de definición es el siguiente:

definición Almacen

usa booleano,

producto {Una de sus operaciones es CreaP: entero entero → producto}

tipo caja; {estructura que guarda información sobre el código y el tipo de un producto, y que se identifica por un número de caja.

Tanto el número de la caja, como el código y el tipo del producto son de tipo entero}

acción CreaCaja (p: producto; n: entero; **var** c: caja);

{Esta acción crea la caja **c** con el número **n**, y la carga con el producto **p**}

acción VerCaja (c: caja);

{Esta acción muestra por pantalla el número identificativo de la caja **c**, así como el código y el tipo del producto que almacena}

función Esta? (c: caja; p: producto): booleano;

{Esta función devuelve Verdad si el producto **p** se encuentra en la caja **c**, y falso en caso contrario}

finDefinición

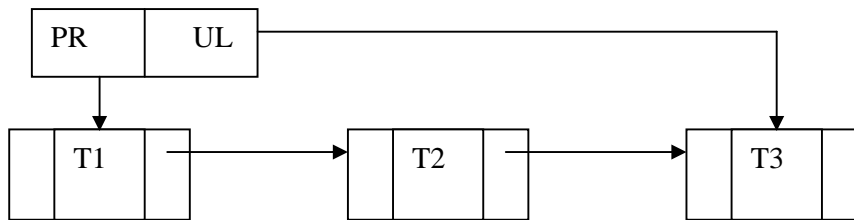
Se pide: Escribir en lenguaje algorítmico un programa de usuario que, basándose en el TAD *almacén*, realice las siguientes operaciones:

1. Crear el fichero de Cajas *MisCajas.dat*, en el que cada registro será una caja identificada por un número secuencial de orden. Los productos que se almacenarán en estas cajas serán los que contenga el fichero de datos secuencial *Stock.dat*. Los registros de este fichero de datos están formados por dos campos: *CodProd* y *TipProd*, ambos de tipo entero, y contienen la información de los distintos productos.
2. Mostrar por pantalla el número de caja, así como el código y tipo del producto cargado en cada una de las cajas almacenadas en el fichero *MisCajas.dat*.
3. Dados el código y tipo de un producto, comprobar si se encuentra en alguna caja del fichero de cajas *MisCajas.dat*.

Cuestión 4

1.5 puntos

Dada una Lista Enlazada de Tablas (LET) como la mostrada en el siguiente *ejemplo*:



donde los datos T1, T2 y T3 son tablas de enteros, y las posiciones de las tablas que no están ocupadas (vacías) tienen almacenado un **cerro (0)**. Considerar la siguiente representación para el tipo LET:

Tipo LET = registro
PR, UL : nodo;
fregistro

nodo = **registro**
T: **tabla** [1..10] de entero;
sig : nodo;
fregistro

Se Pide:

- (a) Escribir, en lenguaje algorítmico, la acción **Repetidos**, que dada una LET, elimina todos los enteros repetidos de cada uno de los nodos de la lista, y devuelve en **r** el número de elementos repetidos que se han eliminado en total en todos los nodos.

acción Repetidos (**var** L: LET; r: entero);

Ejemplo: Una LET con sólo dos nodos, cuya información es T1(0,1, 2, 0, 0, 0, 1, 0, 4, 0) T2(0, 0, 2, 0, 3, 1, 0, 2, 0, 4) quedaría de la siguiente forma, tras procesarla con esta acción:

T1(0, 1, 2, 0, 0, 0, 0, 0, 4, 0) T2(0, 0, 2, 0, 3, 1, 0, 0, 0, 4)

y en **r** se devolvería el valor 2, ya que se ha eliminado 1 elemento de T1 y otro en T2

- (b) Escribir, en lenguaje algorítmico, la acción **L123**, descrita a continuación en un pseudocódigo de muy alto nivel:

acción L123 (**var** L: LET; d: entero)

var C: conjunto de enteros

{ las operaciones válidas sobre el tipo *conjunto de enteros* son:
poner(C, d) y *esta?*(d, C) }

m: entero

L1- Calcular el entero mayor de los existentes en L y almacenarlo en *m*

L2- Guardar en C todos los enteros distintos almacenados en L, que no sean 0

L3- Eliminar el entero *m* de todos los nodos de L menos del primer nodo en el que lo encuentre

faccion

IMPORTANTE:

- Cada línea (L1, L2 y L3) de la acción **L123** debe ser implementada mediante una función o una acción.
- No sólo se valorará el correcto funcionamiento de la acción **L123**. Se tendrá muy en cuenta la correcta metodología de programación utilizada para su implementación.