T−2. Árboles B. Problemas de dificultad ascendente.

- Dado un árbol B con tamaño de página 4 que inicialmente se encuentra vacio, indicar para cada uno de los siguientes casos, el contenido y la estructura de dicho árbol:
- Se insertan las claves 10, 20, 30, 40.
- A continuación, se insertan las claves 15, 70, 70.
- A continuación, se insertan las claves 35, 70, 80.
- A continuación, se insertan las claves 55, 85, 90.
- A continuación, se insertan las claves 65, 95, 96.
- A continuación, se inserta la clave 84.
- Finalmente, se borra la clave 60.
- Dado un árbol B con la siguiente declaración:

```
Const

N = 2; (* Orden del árbol B *)

DosN = 2 * N; (* Tamaño de página *)

Type

tClave = 1..Maxint;

RangoClaves = 1..DosN;

RangoDescen = 0.. DosN;

tArbolB = !tPagina;

tPagina = Record

NumClaves : RangoClaves;

Claves : Array[RangoClaves] of tClave;

Descen : Array[RangoDescen] of tArbolB;

end;
```

Se pide implementar un procedimiento que imprima las claves del árbol B de menos a mayor. Supóngase que la clave es imprimible directamente.

- Dada la siguiente declaración de árbol B:

```
Type

tArbolB = !Pagina;

Pagina = Record
```

NumClaves : **Integer**;

Clave : **Array**[1..N] **of** tClave;

Descen : **Array**[0..N] **of** tArbolB;

end;

Codificar una función en Pascual que, recibiendo como parámetro un árbol B perteneciente al tipo anterior, devuelva un puntero a aquel subárbol (de entre los que cuelgan de la página raíz) que posea el mayor número de claves.

OBSERVACIONES:

_ No se permite la utilización de ninguna estructura de datos auxiliar.

_ Sólo se permite la realización de un recorrido en el árbol.

_ En caso de que varios subárboles posean el mayor número de claves, la función devolverá un puntero a aquel cuya suma sea la mayor.

- Dada la siguiente declaración:

Const

N = ;

DosN = ;

Type

tClave = **Integer**;

tArbol = !Pagina;

tPagina = **Record**

NumClaves : 1..DosN;

Claves : **Array**[1..DosN] **of** tClave;

Descend : **Array**[1..DosN] **of** tArbol;

end;

Se pide codificar una **función recursiva** que, recibiendo un puntero del tipo `tArbol' que apunta a un árbol multicamino, que se sabe que es de búsqueda, determine si dicho árbol cumple las características de un árbol B. Cada página del árbol sólo podrá ser visitada una vez.

- Dado un árbol B con la siguiente declaración:

Const

N = 4;

DosN = 2 * N;

Type

TClave = 1..**Maxint**;

RangoClaves = 1..DosN;

RangoDescen = 0.. DosN;

TArbol = !TPagina;

TPagina = **Record**

NumClaves : RangoClaves;

Claves : **Array**[RangoClaves] **of** TClave;

Descen : **Array**[RangoDescen] **of** TArbol;

end;

en el cual pueden existir claves repetidas, sin saber donde se insertaron, se pide codificar un algoritmo que determine la existencia o no de claves repetidas en el árbol B.

NOTA: No se admitirán como válidas aquellas soluciones que realicen más de un recorrido del árbol o utilicen una estructura auxiliar.

- Dada la siguiente declaración de árbol B:

CONST

N = 4;

DosN = 2 * N;

TYPE

TClave = 1..**Maxint**;

RangoClaves = 1..DosN;

RangoDescen = 0..DosN;

TArbol = !TPagina;

TPagina = **Record**

NumClaves : RangoClaves;

Claves : **Array**[RangoClaves] **of** TClave;

Descen : **Array**[RangoDescen] **of** TArbol;

end;

Se pide:

Dados dos árboles A1 y A2, codificar un algoritmo en Pascual que determine si el árbol A1 se encuentra incluido dentro del árbol A2 (un árbol A1 está incluido en otro A2 cuando todas las claves de A1 están en A2 y además están distribuidas topologicamente del mismo modo).

NOTA: No se admitirán como válidas aquellas soluciones que realicen más de un recorrido del árbol o utilicen una estructura auxiliar.