ASIGNATURA		CURSO	CALIFICACIÓN
ESTRUCTURA DE DATOS Y DE LA INFORMACIÓN		**2003 / 2004**	
TITULACIÓN	GRUPO	CONVOCATORIA	
		ORDINARIA-JUNIO	
APELLIDOS		NOMBRE	

EJERCICIO 1 (3 PUNTOS)

A) Dado el tipo abstracto de datos (TAD) `tLista` que sirve para representar listas de enteros, del que sólo se conoce la parte de la interfaz siguiente:

```
type
   tInfo=integer;
   tLista=...

   function esListaVacia(l:tLista): boolean;

   function cabeza(l:tLista):Integer;
   {Objetivo: Devuelve el entero almacenado en el primer nodo     }
   {          de la lista                                         }
   {Precond:  La lista no está vacía                              }

   function resto(l:tLista):tLista;
   {Objetivo: Devuelve la sublista resultante de eliminar         }
   {          el primer nodo de la lista                          }
   {Precond:  La lista no está vacía                              }
```

Se pide: Escribir un procedimiento o función recursivo que devuelva el mayor entero de una lista desordenada de enteros positivos.

B) Dado el tipo abstracto de datos (TAD) `tArbolBin` que sirve para representar árboles binarios de enteros, del que sólo se conoce la parte de la interfaz siguiente:

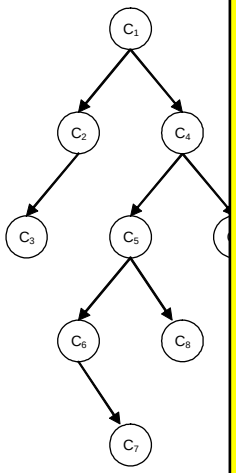
```
type
   tInfo = integer;
   tArbolBin = …

   function EsVacio   (a: tArbolBin): boolean;
   function Raiz      (a: tArbolBin): tInfo;
   function RamaIzqda (a: tArbolBin): tArbolBin;
   function RamaDcha  (a: tArbolBin): tArbolBin;
```

Se pide: Definir una función que devuelva el número de nodos internos (nodos no hoja) de un árbol,

```
function numNodosInternos (a: tArbolBin): integer;
```

EJERCICIO 2 (3 PUNTOS)

A) Sea el árbol AVL siguiente:



resultado de insertar el nodo con contenido … el contenido de cada nodo (incluida la clave). Dete… …ado, y en ese caso reestructurarlo, indicando los fa… …y la rotación efectuada, dibujando el estado del árbol…

B) En un sorteo de lotería se cuenta con dos bomb… …los números y el segundo a los premios. Para alma… …nios correspondientes se busca una estructura de … …rma eficiente el premio asociado a un número en con… …a es la más adecuada, y por qué.

C) Supongamos las siguientes implementaciones de… …ada, dinámica con dobles enlaces, dinámica circular … …icar para las operaciones que se detallan a continuaci… …nes anteriores es más eficiente.

Operación
`Ultimo (Lista)-> Posición`
`Anterior (Lista, Posición)-> Posición`
`Siguiente (Lista, Posición)-> Posición`
`Insertar (Lista, Posición, Elemento)->`
`Acceso al n-ésimo elemento`

EJERCICIO 3 (4 PUNTOS)

Consideremos que un conjunto es una colección de elementos todos del mismo tipo sin duplicidades. Sea entonces una implementación del tipo abstracto de datos *TAD Conjunto* de números naturales.

La especificación informal de operadores para este TAD es la siguiente (Conjunto es de tipo *tConjunto*; elemento es de tipo *tInfo*).

ConjuntoVacio (Conjunto) → Conjunto
{Objetivo: Crear un conjunto vacío
Salida: Un conjunto vacío}

EsConjuntoVacio (Conjunto) → Boolean
{Objetivo: Determinar si un conjunto está vacío
Entrada: *Conjunto*
Salida: *true* si el conjunto está vacío, *false* en caso contrario.
PreCond: El *Conjunto* debe estar inicializado }

Insertar (Conjunto, elemento) → Conjunto
{Objetivo: Añadir al conjunto un elemento
Entrada: *Conjunto, Elemento*
Salida: *Conjunto* con el nuevo elemento si no pertenecía al conjunto
PreCond: El *Conjunto* debe estar inicializado, el *elemento* no pertence al *Conjunto* y se supone memoria suficiente}

Pertenece (Conjunto, elemento) → boolean
{Objetivo: Indica si el *elemento* está en el *Conjunto*
Entrada: *Conjunto, Elemento* a buscar
Salida: *true* si el *elemento* está en el *Conjunto*, *false* en caso contrario
PreCond: El *Conjunto* debe estar inicializado }

Borrar (Conjunto, elemento) → Conjunto
{Objetivo: Elimina el *elemento* del *Conjunto*
Entrada: *Conjunto, Elemento* a buscar
Salida: El *conjunto* sin el *elemento*
PreCond: El *elemento* debe pertenecer al *Conjunto*}

Intersección (Conjunto, Conjunto) → Conjunto
{Objetivo: Realiza la intersección de dos conjuntos
Entrada: Dos *Conjuntos*
Salida: Un nuevo *conjunto* resultado de la intersección de los conjuntos de entrada
PreCond: Los conjuntos deben estar inicializados y se supone memoria suficiente}

Se pide:

A) Realizar en lenguaje Pascal la definición de tipos del *TAD Conjunto*. La estructura será una **lista ordenada** y se deberá justificar la elección de la implementación (estática o dinámica).

B) Realizar la implementación de las operaciones definidas anteriormente para el *TAD Conjunto*, usando la definición de tipos usada en el apartado **(A)**.

SOLUCIONES:

EJERCICIO 1 (3 PUNTOS)

A) 1'5 punto

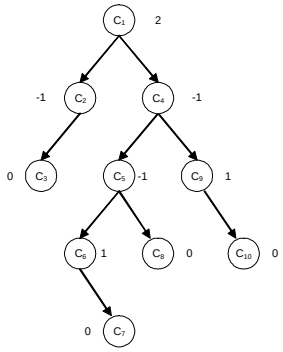
```
function mayor (L:tLista): integer;
begin
      if EsListaVacia(L)
      then mayor:=0
      else mayor:=maximo(cabeza(L), mayor(re
end;
```

B) 1'5 puntos

```
function numnodosinternos (arbol: tarbolbin)
begin
  if EsVacio(arbol)
  then numnodosinternos:=0
  else
  if EsVacio(RamaDcha(arbol)) and EsVacio(Ra
        then numnodosinternos:=0
        else numnodosinternos:= 1 + numnodos
                                  + numnodos
end;
```

EJERCICIO 2 (3 PUNTOS, 1 CADA EJERCICIO)

A) El árbol AVL siguiente está desequilibrado:



Se r
con
DI,

1
0
C10
0

B) Estructura para almacenar y buscar de forr
estructura debe estar ordenada para permitir
estructuras ordenadas la de mayor eficiencia en
al tener los nodos equilibrados y las clav
comparaciones necesarias para encontrar una cl
números premiados.

La
las
ue
de
los

C)

Operación	Implementaciones más eficientes
Ultimo (Lista)-> Posición	Circular, estática
Anterior (Lista,Posición)-> Posición	Estática, dinámica con dobles enlaces
Siguiente (Lista, Posición)-> Posición	Todas
Insertar (Lista, Posición, Elemento)-> Lista	Dinámicas
Acceso al n-ésimo elemento	Estática

EJERCICIO 3 (4 PUNTOS, TIPOS+ESVACIO+CONJUNTOVACIO=0.75, INTERSECCION=1 Y RESTO 0.75)

Lista ordenada dinámica, puesto que el número de elementos (actualizaciones) es desconocido a priori.

```
Unit TADConjunto;

INTERFACE

const nulo = nil;
type
    tInfo = 1..MaxInt;
    tPos = ^tNodo;
    tNodo = record
      elemento: tInfo;
      sig : tPos;
    end;
    tConjunto = tPos;
```

IMPLEMENTATION

```
Procedure ConjuntoVacio (var C: tConjunto);
{Objetivo:  Crear un conjunto vacío
Salida:     Un conjunto vacío}
begin
    C:= nulo;
end;


Function EsConjuntoVacio (C: tConjunto): boolean;
{Objetivo:  Determinar si un conjunto está vacío
Entrada:    Conjunto
Salida:     true si el conjunto está vacío, false en caso contrario.
PreCond:    El Conjunto debe estar inicializado }
begin
    EsConjuntoVacio:= (c=nulo);
end;
```

```
Procedure Insertar (var C: tConjunto; e: tEl
{Objetivo:  Añadir al conjunto un elemento
Entrada:    Conjunto, Elemento
Salida:     Conjunto con el nuevo elemento s
PreCond:    El Conjunto debe estar iniciali                    ece
            al Conjunto y se supone memoria
var
    aux, nuevo: tPos;
begin
    new(nuevo);
    nuevo^.elemento:= e;
    nuevo^.sig:= nulo;
    if EsConjuntoVacio(C)
    then
      C:= nuevo
    else begin                              {Loc
      anterior:= nulo;
      aux:= C;
      while  (e > aux^.elemento) and (aux^.s
        begin
            anterior:= aux;
            aux:= aux^.sig;
        end;

      if e > aux^.elemento then
            anterior:= aux;

      if anterior= nulo
      then begin
            nuevo^.sig:= C;
            C:= nuevo
      end else begin
            nuevo^.sig:= anterior^.sig;
            anterior^.sig:= nuevo;
      end;
    end; (* fin del else *)
end; (* fin de insertar *)
```

```
Function Pertenece (C: tConjunto; e: tElemen
{Objetivo:  Indica si el elemento está en el
Entrada:    Conjunto, Elemento a buscar
Salida:     true si el elemento está en                        so
contrario
PreCond:    El Conjunto debe estar inicializ

begin
    if EsConjuntoVacio(C)
    then Pertenece:= false
    else begin
      while (C^.sig <> nulo) and (e > C^.ele
            C:= C^.sig;
      Pertenece:= (C^.elemento = e)
    end;
end;
```

```
Procedure Borrar (var C: tConjunto; e: tElemento);
{Objetivo:  Elimina el elemento del Conjunto
Entrada:    Conjunto, Elemento a buscar
Salida:     El conjunto sin el elemento
PreCond:    El elemento debe pertenecer al Conjunto}
var
   aux, anterior: tPos;
begin
   aux:= C;
   anterior:= nulo;
   while (e <> C^.elemento) do begin
     anterior:= aux;
     aux:= aux^.sig;
   end;
   if anterior = nulo
   then                                              {Borrado en cabeza}
     C:= C^.sig
   else                                        {Borrado en medio o al final}
     anterior^.sig:= aux^.sig;
   dispose(aux);
end;
```

```
Procedure Interseccion (C1, C2: tConjunto; var C: tConjunto);
{Objetivo:  Realiza la intersección de dos conjuntos
Entrada:    Dos Conjuntos
Salida:     Un nuevo conjunto intersección de los conjuntos de entrada
PreCond:    Los conjuntos deben estar inicializados
            y se supone memoria suficiente}

var
   nuevo, ultimo: tPos;
begin
   ConjuntoVacio(C);
   while not (EsConjuntoVacio(C1) or EsConjuntoVacio(C2)) do begin
     if C1^.elemento = C2^.elemento      {Tenemos dos elementos iguales}
     then begin
           new(nuevo);
           nuevo^.elemento:= C1^.elemento;
           nuevo^.sig:= nulo;

           if EsConjuntoVacio(C)  {Comprobamos si el conjunto está vacio}
           then C:= nuevo;
           else ultimo^.sig:= nuevo;
           ultimo:= nuevo;           {Actualizamos el puntero al último}

           C1:= C1^.sig;
           C2:= C2^.sig;
     end
     else
           if C1^.elemento < C2^.elemento
           then C1:= C1^.sig
           else C2:= C2^.sig
   end;
end;
```