ASIGNATURA		CURSO	CALIFICACIÓN
ESTRUCTURA DE DATOS Y DE LA INFORMACIÓN		**2005 / 2006**	
TITULACIÓN	GRUPO	CONVOCATORIA	
		EXTRAORDINARIA - DIC	
APELLIDOS		NOMBRE	

EJERCICIO 1 (3 PUNTOS)

A) Dados los siguientes supuestos prácticos decidir cuál es la **MEJOR estructura de datos** y su **MEJOR implementación** (estática o dinámica) para resolver el problema, **justificando** la respuesta:

1. Una empresa de seguridad necesita implementar un programa de monitorización cíclica de cámaras de videovigilancia que dado un conjunto de identificadores de cámaras, se conecte durante toda la noche a cada una de ellas en secuencia para mostrar su imagen durante breves instantes. La secuencia de cámaras debe ser siempre la misma, para no despistar a los agentes de la sala de control. ¿Qué estructura de datos debe utilizarse para almacenar los identificadores de las cámaras y saber cuál de ellas se debe mostrar en cada momento?

2. Una empresa de reparto de empanadas a domicilio precisa ampliar su sistema informático con un programa que asigne cada nuevo pedido al motorista que lleva más tiempo desocupado. Normalmente hay 10 motoristas trabajando, pero los fines de semana y los días en que se retransmite fútbol europeo por la televisión se cuenta con más personal. Además, cada vez que un motorista regresa a la empresa después de servir un pedido, introduce su identificador en el sistema informático para indicar que ha quedado libre ¿Qué estructura de datos utilizaremos para almacenar los identificadores de los motoristas disponibles?

3. Un popular programa de televisión pretende realizar un sorteo utilizando el siguiente método: un notario proporcionará un listado con los números de teléfono de aquellos telespectadores que se han apuntado al sorteo. De entre esos números se elegirá uno al azar y se llamará. Si responde correctamente a una pregunta, se le asignará el premio; en caso contrario, se repetirá el proceso con el número anterior del listado, con el siguiente, con el anterior del anterior, con el siguiente del siguiente, y así sucesivamente hasta que alguien responda correctamente o se termine el listado. ¿Qué estructura de datos deberemos utilizar para almacenar los números de teléfono?

B) Contestar **Verdadero o Falso** a las siguientes preguntas, **justificando** la respuesta:

1. El único criterio para elegir entre una implementación estática o dinámica es el consumo de memoria previsto.

2. La eliminación de una clave en un árbol binario de búsqueda balanceado obliga siempre a realizar al menos una rotación.

3. En un árbol en montículo, cada nuevo elemento se inserta siempre en la posición ocupada por la raíz.

4. Un recorrido inorden de un árbol AVL devuelve la secuencia ordenada de claves.

C) Sea una implementación ESTÁTICA del TAD Lista en la que se trabaja con un número máximo de elementos establecido por la constante MAX. Explica los errores que presentaría el código siguiente respecto a su funcionamiento y/o el cumplimiento de la especificación de la operación (objetivo, entradas, salidas y precondiciones):

```
function insertarPrimero (var L: tLista; d : tDAto): boolean;
{ Objetivo: insertar el elemento d en la primera posición de la lista
  Entrada: la lista L y el dato d a insertar
  Salida: la lista L modificada y true si la inserción se ha realizado
        con éxito. En caso contrario, L sin modificar y false.
  PreCD: la lista está correctamente inicializada }

var
  i : tPos;
begin
  for i:=1 to L.tamano do
    L.datos[i+1] := L.datos[i];
  L.datos[1] := d;
end;
```

EJERCICIO 2 (4 PUNTOS)

Una bolsa puede ser definida como una colección de elementos entre los que no se establece ninguna relación de orden, tal y como ocurre en un conjunto. Una vez que introducimos un elemento en la bolsa no podremos decir en qué orden va a quedar dicho elemento con respecto al resto de elementos de la bolsa. De la misma forma, al extraer un elemento de la bolsa todos tienen la misma posibilidad de ser extraídos.

El TAD bolsa está compuesto de dos tipos de datos: tBolsa (que representa a la bolsa) y tInfo (que representa el tipo de un elemento de la bolsa, en nuestro caso será integer); y su comportamiento se define a través de las siguientes funciones disponibles en su interfaz:

- bolsaVacia () → tBolsa — Crea una bolsa vacía
- esBolsaVacia (tBolsa) → boolean — Determina si la bolsa está vacía
- insertar (tBolsa, tInfo) → tBolsa — Inserta un elemento (tInfo) en la bolsa.
- extraer (tBolsa) → tBolsa, tInfo — Extrae aleatoriamente un elemento de la bolsa
  PreCD: La bolsa no está vacía.
  PostCD: Se elimina el elemento extraído
- vaciarEn (tBolsa, tBolsa) → tBolsa, tBolsa — Vacía en la primera bolsa el contenido de la segunda, quedando ésta última vacía.
- eliminar (tBolsa) → tBolsa — Elimina todos los elementos de la bolsa
- tamano (tBolsa) → integer — Devuelve el número de elementos de la bolsa

Se pide:

A) **Definición de tipos** para una implementación **estática** (de máximo 1000 elementos) y una implementación **dinámica** de la cola.

B) Implementar la función **extraer** para la implementación **estática y dinámica**.

C) Implementar la función **vaciarEn** para la implementación **estática y dinámica**.

En los apartados B y C anteriores se valorará que las funciones no sólo cumplan con sus especificaciones, **sino que su implementación sea lo más eficiente posible**.

D) Vamos a utilizar el TAD bolsa para implementar el funcionamiento de un mini sorteo de lotería de navidad. En este sorteo partimos de 1000 boletos y de los siguientes premios (no existen premios por terminaciones):

1 premio gordo de 300 mil euros
1 segundo premio de 200 mil euros
1 tercer premio de 100 mil euros
4 cuartos premios de 50 mil euros
50 pedreas de 10 mil euros

El sorteo consiste en dos bombos, uno con todos los números y otro con los premios. Se van sacando números de ambos bombos y colocando en una tabla de premios que asocia un número con su premio. El procedimiento continúa hasta que el bombo de premios queda vacío.

Se pide, por tanto, implementar la función "loteriaNavidad→tTablaPremios", que se encarga de generar los bombos, realizar el sorteo y almacenar los resultados en una tabla de premios donde se inserte el número y el premio correspondiente (será necesario también definir el tipo de la tabla de premios).

NOTA: Para los apartados B, C y D se pueden utilizar todas las funciones definidas en la interfaz del TAD.

EJERCICIO 3 (3 PUNTOS)

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

```
type
     tArbolBin = ...

function EsArbolVacio (a:tArbolBin):boolean;
function Raiz (a:tArbolBin):integer;
function RamaIzda (a:tArbolBin):tArbolBin;
function RamaDcha (a:tArbolBin):tArbolBin;
```

A) En un árbol binario de búsqueda (A) pretendemos determinar la clave cuyo valor es inmediatamente inferior al de la clave almacenada en el nodo de la raíz. Se devolverá el valor -1 en caso de que no exista la clave buscada

B) Pretendemos determinar si dos árboles binarios son **Iguales**, a través de la siguiente función que devolverá un valor verdadero o falso indicando el resultado de la comparación. ¿Es correcta? Si no es así realizar las correcciones oportunas.

```
Function Iguales (a1,a2: tArbolBin): boolean;
begin
   if EsArbolVacio(a1)and EsArbolVacio(a2)
   then    Iguales:=true
   else    Iguales:= (Raiz(a1)= Raiz(a2)) and
                  Iguales (RamaIzda(a1),RamaIzda(a2)) and
                  Iguales (RamaDcha(a1), RamaDcha(a2));
end;
```

Soluciones

Ejercicio 1:

A)

1. SOLUCION: lista circular, cola con implementación circular. En principio la implementación estática parece la más recomendable ya que se conoce el número de cámaras y se supone que no se van a añadir o eliminar cámaras durante la vigilancia.
2. SOLUCIÓN: una cola (dinámica)
3. SOLUCIÓN: lista doblemente enlazada dinámica (no se sabe a priori cuánta gente se apuntará). Sin embargo, otra explicación es posible: ya que el listado lo proporciona un notario de antemano, el número es conocido, por eso estático.

B)

1. SOLUCIÓN: Falso. Además es necesario valorar la frecuencia de actualización de los datos.
2. SOLUCIÓN: Falso. Sólo cuando el árbol se desequilibre es necesario efectuar una reestructuración.
3. SOLUCIÓN: Falso. La clave se inserta en el último nivel (como una hoja) lo más a la izquierda posible del árbol, y luego se coloca comparándola recursivamente con la clave de su padre (dependiendo de si el árbol es min o max, la clave del padre tendrá que ser menor o mayor)
4. SOLUCIÓN: Verdadero. El recorrido inorden es: Hijo izquierdo, Raiz, Hijo Derecho, y en un AVL, que es un Árbol Binario de Búsqueda, las claves se encuentran ordenadas, es decir, se cumple para todos los nodos del árbol que el hijo izquierdo es menor (mayor) que la raíz, y ésta a su vez menor (mayor) que el hijo derecho.

C)

SOLUCIÓN:

```
begin
  if L.tamano=MAX then
     insertarPrimero := FALSE
  else begin
    for i:=L.tamano downto 1 do
      L.datos[i+1] := L.datos[i];
    L.datos[1] := d;
    L.tamano:=L.tamano+1;
    insertarPrimero := TRUE
  end
end;
```

* El bucle no es descendente
* No devuelve nada
* No comprueba si se acaba el array
* No suma 1 al tamaño final

Ejercicio 2:

A) Definición de tipos

```
(*** ESTÁTICA ***)


CONST
    max=100;

TYPE
    tInfo  = integer;
    tBolsa = record
               elementos: array[1..max] of tInfo;
               ultimo:integer;
             end;


(*** DINÁMICA ***)


TYPE
    tInfo  = integer;
    tBolsa = ^tNodo;
    tNodo  = record
               info:tInfo;
               sig:tBolsa;
             end;
```

La solución más eficiente en la implementación dinámica consistiría en definir la bolsa como un registro en el que se almacene un puntero al primer tNodo de la bolsa, el tamaño de la bolsa (para que la función 'tamaño' no tenga que recorrer la bolsa entera) y un puntero al último elemento (para hacer eficiente 'vaciarEn').

B) Función extraer

```
(*** ESTÁTICA ***)


FUNCTION extraer(var b:tBolsa):tInfo;
var
  num,i:integer;
begin
  num:=random(b.ultimo)+1;
  extraer:=b.elementos[num];
  if num <> b.ultimo
    then b.elementos[num]:=b.elementos[b.ultimo];
  b.ultimo:=b.ultimo-1;
end;
```

```
(*** DINÁMICA ***)

FUNCTION extraer(var b:tBolsa):tInfo;
var
   num,i:integer;
   pos,ant:tBolsa;
begin
  num:=random(tamano(b));
  pos:=b;

  for i:=1 to num do
    begin
      ant:=pos;
      pos:=pos^.sig;
    end;
  extraer:=pos^.info;
  if (pos=b) then //si es el primer elemento
    begin
      b:=b^.sig;
      dispose(pos);
    end
  else
    begin
      ant^.sig:=pos^.sig;
      dispose(pos)
    end;
 end;
```

C) Función vaciarEn

```
(*** ESTÁTICA ***)

PROCEDURE vaciarEn(var Bdestino:tbolsa; var Borigen:tBolsa);
var
   i:integer;
begin
  for i:=1 to Borigen.ultimo do
    begin
      Bdestino.ultimo:=Bdestino.ultimo+1;
      Bdestino.elementos[Bdestino.ultimo]:=Borigen.elementos[i];
    end;
  Borigen.ultimo:=nulo;
 end;

(*** DINÁMICA ***)

PROCEDURE vaciarEn(var Bdestino:tBolsa; var Borigen:tBolsa);
var
  ultimo:tBolsa;
begin
  if esBolsaVacia(Bdestino)then
    Bdestino:=Borigen;
  Else if not esBolsaVacia(Borigen)
  then  begin
      ultimo:=Bdestino;
      while ultimo^.sig<>nulo do
        ultimo:=ultimo^.sig; //se busca el ultimo elemento de Borigen
      ultimo^.sig:=Borigen;  //se enlaza Borigen al final de Bdestino
  end;
  (* En caso de bolsa vacia Borigen, no hacemos nada *)
  Borigen:=nulo;
end;
```


D) Función loteriaNavidad

```
CONST
   NUM_PREMIOS = 57;
TYPE
   tPremio = record
      numero, premio: integer;
   end;
   tTablaPremios = array [1..NUM_PREMIOS] of tPremio;

function loteriaNavidad: tTablaPremios;
var
  numeros, premios: tBolsa;
  i, numPremios, numero, premio:integer;
  tabla: tTablaPremios;
begin
  // Llenamos el bombo de números y actualizamos
  // la tabla de premios
  numeros:=bolsaVacia;

  for i:=0 to MAX_NUM do // MAX_NUM = 1000
    begin
      insertar(numeros, i);
      tabla[i]:=0;
    end;

  // Llenamos el bombo de premios
  premios:=bolsaVacia;

  // Gordo
  insertar(premios, 300);

  // 2º premio
  insertar(premios, 200);

  // 3er premio
  insertar(premios, 100);

  // 3er premio
  insertar(premios, 100);

  // 4os premios
  for i:=1 to 4 do
    insertar(premios, 50);

  // Pedreas
  for i:=1 to 50 do
    insertar(premios, 10);

  // Realizamos el sorteo
  numPremios := tamano(premios);
  for i:=1 to NUM_PREMIOS do
    begin
      numero:=extraer(numeros);
      premio:=extraer(premios);
      tabla[i].numero:= numero;
      tabla[i].premio:= premio;
    end;

  // vaciamos los numeros restantes y devolvemos la tabla
  eliminar(numeros);
  loteriaNavidad:=tabla;
end;
```

Ejercicio 3:

A)

SOLUCIÓN:

```
procedure ClaveMenor (A: tArbolBin);
var B: tArbolBin;
begin
   if EsArbolVacio(A)
   {esta condición vale tanto para cuando el árbol es vacío
   inicialmente como para detectar cuándo la clave no existe en el
   árbol}
   then ClaveMenor:=-1
   else    if EsArbolVacio(RamaIzda(A))
           then ClaveMenor:=-1
           else begin
                   B:= RamaIzda(A);
                   While not EsArbolVacio(RamaDcha(B)) do
                           B:= RamaDcha(B);
                   ClaveMenor:= Raiz(B);
           End;
End;
```

B)

SOLUCIÓN:

```
Function Iguales (a1,a2: tArbolBin): boolean ;
begin
   if EsArbolVacio(a1)and EsArbolVacio(a2)
   then    Iguales:=true
   else    if EsArbolVacio(a1) or EsArbolVacio(a2)
           then Iguales:=false
           else Iguales:= (Raiz(a1)= Raiz(a2)) and
                     Iguales (RamaIzda(a1),RamaIzda(a2)) and
                     Iguales (RamaDcha(a1), RamaDcha(a2));
end;
```