UNIVERSIDAD PONTIFICIA DE SALAMANCA EN MADRID
Facultad de Informática
Escuela Universitaria de Informática
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software

Soluciones propuestas al examen

ASIGNATURA:	**Fundamentos de Programación II**	CÓDIGO:	**113**
CONVOCATORIA:	**Junio de 2004**	PLAN DE ESTUDIOS:	**2000 / 2002**
ESPECIALIDAD:		CURSO:	**1º**
TURNO:	**Mañana**	CURSO ACADÉMICO:	**2003/2004**
CARÁCTER:	**Cuatrimestral (2º cuatrimestre)**	PROGRAMA:	**Ingeniería en Informática / Ingeniería Técnica en Informática**
DURACIÓN APROXIMADA:	**2 horas y media**		

Preguntas teórico-prácticas

1. Colas. Concepto de cola. Describa al menos tres formas de implementar colas. Complemente su explicación con un esquema de cómo se haría cada una de esas implementaciones.

Apartado 12.8 del libro de texto

Aplicación
Eligiendo alguna de las tres implementaciones del apartado anterior, codifique un procedimiento que permita insertar un nuevo elemento en una cola. La cola y el elemento a insertar se pasarán como argumentos al procedimiento.

```
procedimiento CInsertar(E/S cola : c ; E TipoElemento : e)
var
   puntero_a nodo : aux
inicio
   reservar(aux)
   aux↑.sig ← nulo
   aux↑.info ←e
   si c.p = nulo entonces
     c.p ← aux
   si_no
      c.f↑.sig ← aux
   fin_si
   c.f ← aux
fin_procedimiento
```

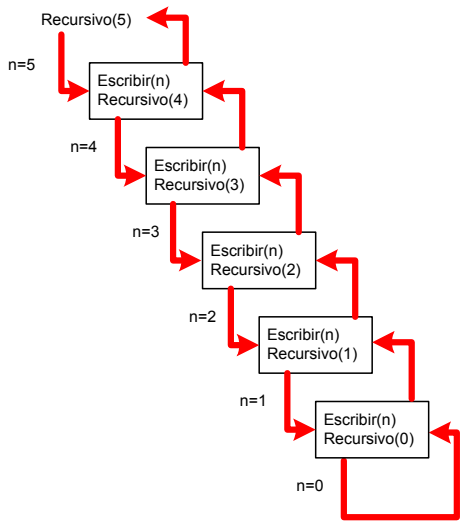
Puntuación: 1,5 puntos

2. Concepto de recursividad. Tipos de recursividad. Elementos de un módulo recursivo.

Apartados 14.1 y 14.2 del libro de texto

Aplicación
Codifique un procedimiento recursivo que escriba los números entre n y 1, siendo n un valor pasado como argumento. Realice un traza del procedimiento codificado para n=5.

```
procedimiento Recursivo(E entero : n)
inicio
   si n <> 0 entonces
      escribir(n)
      Recursivo(n-1)
   fin_si
fin_procedimiento
```



Puntuación: 1,5 puntos

3. Archivos directos con transformación de clave. Utilidad de las funciones hash. Explique por que es necesaria la utilización de funciones de transformación de clave. Problemas en la utilización de funciones hash.

Apartados 9.12.1, 9.12.2 y 9.12.3 del libro de texto

Aplicación
Se tiene creado un archivo directo por transformación de claves, siendo su clave primaria el campo numérico `Codigo`. El archivo va a contener un total de 1000 registros, reservándose 250 para el tratamiento de las colisiones. Codifique una función que busque un registro a partir de un código que se pasa como argumento. La función deberá devolver el número de registro donde se encuentre la clave o -1 en el caso de que no se encuentre.

Puntuación: 1,5 puntos

```
const
   MaxReg = 1000
   UltimoRegistro = 1250
tipos
   registro = TipoRegistro
      entero : código
      //Resto de campos
      ...
      entero : estado //1=ocupado
   fin_registro
   archivo_d de TipoRegistro = TipoArchivo

entero : función Buscar(E/S TipoArchivo : A; E TipoRegistro : R)
//El tratamiento de colisiones colocará a los sinónimos,
//lo más cerca posible del la posición Hash del registro
var
   TipoRegistro : RAux
   entero : NRR
inicio
   NRR ← hash(R)
   leer(A,NRR,RAux)
   si (RAux.codigo = R.Codigo) entonces
      devolver(NRR)
   si_no
      si RAux.estado n<> 0 entonces
         repetir
            NRR ← NRR mod UltimoRegistro + 1
            leer(A,NRR,RAux)
         hasta_que (RAux.codigo = R.Codigo) o RAux.estado = 0
      fin_si
   fin_si
   si (RAux.codigo = R.Codigo) entonces
      devolver(NRR)
   si_no
      devolver(-1)
   fin_si
fin_procedimiento
```

Preguntas prácticas

Una lista simplemente enlazada almacena los datos de una serie de personas. Por cada persona se guarda su DNI, su nombre, edad y su sexo (H para hombres y M para mujeres). La lista ya está creada y los elementos están ordenados de forma ascendente por nombre.

1. Escriba la declaración de todas las estructuras de datos necesarias para realizar las acciones de los puntos siguientes.

```
tipos
   registro = TipoElemento
      cadena : DNI, nombre
      entero : edad
      carácter : sexo
   fin_registro
```

```
   puntero_a nodoLista = lista

   registro = nodoLista
      TipoElemento : info
      lista : sig
   fin_registro

   puntero_a nodoArbol = árbol

   registro = nodoArbol
      TipoElemento : info
      árbol : hizq, hder
   fin_registro
```

Puntuación: 0,5 puntos

2. Codifique un módulo que elimine de la lista a las personas menores de 18 años.

```
procedimiento EliminarMenores18(E/S lista : l)
var
   lista : act, ant
inicio
   act ← l
   mientras act <> nulo hacer
      si act↑.info.edad < 18 entonces
         //Si es el primero de la lista, se elimina el elemento l
         si act = l entonces
            LBorrar(l)
         si_no
            //En caso contrario se elimina el nodo al que apunta anterior
            LBorrar(ant↑.sig)
         fin_si
      fin_si
      ant ← act
      act ← act↑.sig
   fin_mientras
fin_procedimiento

procedimiento LBorrar(E/S lista : l)
var
   lista : aux
inicio
   si l = nulo entonces
      //Error, la lista está vacía
   si_no
      aux ← l
      l ← l↑.sig
      liberar(aux)
   fin_si
fin_procedimiento
```

Puntuación: 1,5 puntos

3. Codifique un módulo que permita copiar en una nueva lista todos los hombres. La nueva lista también estará ordenada de forma ascendente por nombre.

```
procedimiento CopiarHombres(E lista : l; E/S lista : listaHombres)
//Se hace un procedimiento recursivo para que se mantenga el orden de la lista
inicio
   si l = nulo entonces
      //Si se llega al fina, se crea la lista de hombrer
      listaHombres ← nulo
   si_no
      CopiarHombres(l↑.sig, listaHombres)
      si l↑.info.sexo = 'H' entonces
         LInsertar(listaHombres, l↑.info)
```

```
      fin_si
   fin_si
fin_procedimiento

procedimiento LInsertar(E/S lista : l ; E TipoElemento : e)
var
   lista : aux
inicio
   reservar(aux)
   aux↑.info ← e
   aux↑.sig ← l
   l ← aux
fin_procedimiento
```

Puntuación: 1,5 puntos

4. Utilizando un árbol binario de búsqueda como estructura de datos auxiliar, realice un listado de todos lo elementos de la lista ordenados de forma ascendente por edad.

```
procedimiento listado(E lista : l)
var
   árbol : a
inicio
   //Se copian todos los elementos de la lista a un árbol
   a ← nulo
   mientras l <> nulo hacer
      InsertarEnÁrbol(a, l↑.info)
      l ← l↑.sig
   fin_mientras
   //Se recorre el árbol en inorden
   RecorrerInOrden(a)
fin_procedimiento

procedimiento InsertarEnÁrbol(E/S árbol :a ; TipoElemento : e)
inicio
   si a = nulo
      //Hemos llegado a una hoja. Insertamos
      reservar(a)
      a↑.info ← e
      a↑.hizq ← nulo
      a↑.hder ← nulo
   si_no
      si a↑.info.edad > e.edad then
         InsertarEnÁrbol(a↑.hizq,e)
      si_no
         InsertarEnÁrbol(a↑.hder,e)
      fin_si
   fin_si
fin_mientras

procedimiento RecorrerInOrden(E arbol: a)
inicio
   si a <> nulo entonces
      RecorrerInOrden(a↑.hizq)
      //Procesar elemento raíz
      escribir(a↑.info.DNI)
      escribir(a↑.info.nombre)
      escribir(a↑.info.edad)
      escribir(a↑.info.sexo)
      RecorrerInOrden(a↑.hder)
   fin_si
fin_procedimiento
```

Puntuación: 2 puntos