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

Cuadernillo de examen

ASIGNATURA:	**Fundamentos de la Programación II**	CÓDIGO:	**113**
CONVOCATORIA:	**Septiembre de 2005**	PLAN DE ESTUDIOS:	**2000/2002**
CURSO:	**1º**	CURSO ACADÉMICO:	**2004-2005**
TURNO:	**Mañana**	PROGRAMA:	**Ingeniería Técnica en Informática** **Ingeniería Informática**
CARÁCTER:	**Cuatrimestral (2º cuatrimestre)**	ESPECIALIDAD:	**Común**
DURACIÓN APROXIMADA:	**2 horas 30 minutos**		

Solución propuesta

Preguntas teórico-prácticas

1. Concepto de pila. Aplicaciones de las pilas. Explique gráficamente al menos dos formas de implementar pilas.

Apartado 12.7 del libro de texto

Aplicación

Diseñe un procedimiento que devuelva una copia de una pila a partir de otra que se pasa como argumento. La pila resultante deberá tener los mismos elementos y en el mismo orden.

```
procedimiento CopiarPila(E/S pila : p,copia)
var
   TipoElemento : e
   Pila : aux
inicio
   //Copia los elementos en orden inverso en una pila auxiliar
   crearPila(aux)
   mientras no EsPilaVacia(p) hacer
      Pop(p,e)
      PInsertar(aux,e)
   fin_mientras
   //Restaura los elementos en la pila copia
   CrearPila(copia)
   mientras no EsPilaVacía(aux) hacer
      Pop(aux,e)
      PInsertar(copia,e)
   fin_mientras
fin_procedimiento
```

Puntuación 1,5 puntos.

2. Archivos directos con transformación de clave. Organización del archivo. ¿Cuándo se produce una colisión? ¿Cómo se pueden resolver las colisiones?

Apartados 9.12.1, 9.12.2 y 9.12.3 del libro de texto

Aplicación

Diseñe un procedimiento que permita insertar un registro en un archivo directo con transformación de clave. El registro a insertar tendrá un campo llamado `clave` con su clave primaria (el resto de campos no son significativos para la aplicación). El número de registros previstos que debe almacenar el archivo será de 1000.

```
procedimiento Alta(E/S tArchivo : A; E tRegistro : R)
var
   tRegistro : RAux
   entero : NRR
inicio
   NRR ← hash(R)
   leer(A,NRR,RAux)
   si RAux.estado = 'O' entonces
      NRR ← ZonaPrincipal
      repetir
          NRR ← NRR + 1
```

```
            leer(A,NRR,RAux)
         hasta_que (RAux.estado <> 'O') o (NRR = MaxReg)
      fin_si
      si RAux.estado = 'O' entonces
         escribir(A,NRR,R)
      si_no
         //Zona de colisiones llena
      fin_si
fin_procedimiento
```

Puntuación 1,5 puntos.

3. Concepto de árbol. Estructura de un nodo de un árbol binario. Explique los conceptos de:

- Nivel de un nodo.
- Altura de un árbol.
- Árbol equilibrado.
- Árbol binario de búsqueda.

Apartados 13.2 y 13.3.1 del libro de texto

Aplicación

Diseñe un procedimiento **recursivo** que permita la inserción en un árbol binario de búsqueda.

```
procedimiento Insertar(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 > e then
         Insertar(a↑.hizq,e)
      si_no
         Insertar(a↑.hder,e)
      fin_si
   fin_si
fin_mientras
```

Puntuación: 1,5 puntos

Preguntas prácticas

Una tienda de discos tiene almacenado su stock en un archivo directo con transformación de claves. El archivo tiene capacidad para 1500 registros y cada registro tiene los siguientes campos:

Nombre del campo	Tipo	Observaciones
Ref	Cadena	Clave primaria con la referencia de disco
Título	Cadena	
Interprete	Cadena	
Género	Cadena	
Estado	Numérico	Contiene un 1 si el registro contiene información válida o un 0 en caso contrario

Se pide:

1. Declarar las estructuras de datos necesarias para realizar todas las operaciones descritas a continuación:

```
const
   MaxReg = 1500
tipos
   registro = rDiscos
```

```
      cadena : Ref, título,intérprete, género
      entero : estado
   fin_registro

   archivo_d de rDiscos = aDiscos

   TipoElemento = rDiscos
   puntero_a nodo = lista
   registro = nodo
      TipoElemento : info
      lista : sig
   fin_registro

   registro = cola
      puntero_a nodo :p,f
   fin_registro
```

Puntuación: 0,5 puntos

2. Desarrollar un procedimiento que copie en una lista enlazada todos los registros del archivo con información válida. La lista resultante estará ordenada por el campo `Género`.

```
procedimiento llenarLista(E aDiscos : A; E/S lista : l)
var
   rDiscos : R
inicio
   abrir(A, lectura, 'DISCOS.DAT')
   leer(A,R)
   l ← nulo   //Se inicializa la lista
   mientras no fda(A) hacer
      si R.estado = 1 entonces
         insertarOrdenado(l,R)
      fin_si
      leer(A,R)
   fin_mientras
   cerrar(A)
fin_procedimiento

procedimiento insertarOrdenado(E/S lista : l; E TipoElemento : e)
var
   lista : act,ant
   lógico : encontrado
inicio
   encontrado← falso
   act ← l
   mientras no encontrado y (act <> nulo) hacer
      si e.género = act↑.info.género entonces
         encontrado ← verdad
      si_no
            ant ← act
            act ← act↑.sig
      fin_si
   fin_mientras
   si act = l entonces
      LInsertar(l,e)
   si_no
      LInsertar(ant↑.sig,e)
   fin_si
fin_prodedimiento

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

```
   l ← aux
fin_procedimiento
```

Puntuación: 1,5 puntos

3. Desarrollar un procedimiento que copia en una cola los elementos cuyo género sea 'Flamenco'.

```
procedimiento copiarFlamenco(E lista : l; E/S cola : c)
inicio
   //Inicializar la cola
   c.p ← nulo
   c.f ← nulo
   mientras l <> nulo hacer
      si l↑.info.género = 'Flamenco' entonces
        CInsertar(c,e)
      fin_si
      l ← l↑.sig
   fin_mientras
fin_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

4. Desarrollar un procedimiento que elimine de la cola aquellos elementos cuyo intérprete sea 'El Lebrijano'. En este procedimiento se deberán utilizar la operaciones básicas para trabajar con colas que también deberán codificarse.

```
procedimiento eliminarAElLebrijano(E/S cola : c)
var
   cadena : primerCD // Referencia del primer CD de la cola
inicio
   si no EsColaVacía(c) entonces
      //Saco la referencia del primer CD de la cola
      Primero(c,e)
      primerCD ← e.Ref
      repetir
         Sacar(c,e)
         si e.intérprete <> 'El Lebrijano' entonces
            //Se inserta al final de la cola
            CInsertar(c,e)
         si_no
            //Si el primer CD ya era de El Lebrijano
            si e.Ref = primerCD entonces
               //El primer CD será ahora el siguiente
               Primero(c,e)
               primerCD ← e.Ref
            fin_si
         fin_si
         Primero(c,e)
      hasta_que e.Ref = primerCD
   fin_si
fin_procedimiento

lógico: función EsColaVacía(E cola : c)
inicio
```

```
   devolver(c.p = nulo) //o (c.f = nulo)
fin_función

procedimiento Primero(E cola : c; E/S TipoElemento : e)
inicio
   si c.p = nulo entonces
      // Error, la pila está vacía
   si_no
      e ← c.p↑.info
   fin_si
fin_procedimiento

procedimiento Sacar(E/S cola : c ; E/S TipoElemento : e)
var
   ptr : aux
inicio
   si c.p = nulo entonces
      // Error, la cola está vacía
   si_no
      e ← c.p↑.info
      aux ← c.p
      c.p ← c.p↑.sig
      liberar(aux)
fin_procedimiento
```

Puntuación: 2 puntos