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 Programación II	CÓDIGO	113
CONVOCATORIA	Parcial de Abril de 2003	PLAN DE ESTUDIOS	2000
ESPECIALIDAD	Común	CURSO	1º
TURNO	Mañana	CURSO ACADÉMICO	2002/2003
CARÁCTER	Cuatrimestral	PROGRAMA	Ingeniería Técnica en Informática
DURACIÓN APROXIMADA			

Pregunta teórica

Ordenación externa

Conteste las siguientes preguntas:

1. ¿Qué métodos de ordenación externa conoce? Explique brevemente cómo realizan la ordenación.
2. ¿Qué métodos de ordenación externa no utilizan arrays?
3. ¿Qué método de partición utiliza el método de ordenación por mezcla directa?

Utilizando el método de ordenación que desee (sin utilizar arrays) realice un seguimiento que permita ordenar un archivo con las siguientes claves indicando cómo quedarían las claves después de cada pasada.

10	21	13	15	18	3	13	24	16	4	98	

A1	10	21	3	13	24	4	98				
A2	13	15	18	16							
A	10	13	15	18	21	3	13	16	24	4	98

A1	10	13	15	18	21	4	98				
A2	3	13	16	24							
A	3	10	13	13	15	16	18	21	24	4	98

A1	3	10	13	13	15	16	18	21	24		
A2	4	98									
A	3	4	10	13	13	15	16	18	21	24	98

Puntuación: 1,5 puntos

Pregunta práctica

Se tiene una frase almacenada en una lista enlazada de forma que cada palabra es un elemento. La lista incluirá también como elementos separados los signos de puntuación (sólo se considerarán los puntos y las comas) y los espacios en blanco.

Se desea hacer un programa que:

1. Elimine de la lista los espacios en blanco.

Declaraciones de tipos de datos

```
tipos
   cadena = TipoElemento //El tipo de elementos del conjunto
   puntero_a nodo = lista //El conjunto sería una lista enlazada
   registro = nodo
      TipoElemento : info
      lista : sig
    fin_registro

procedimiento BorrarBlancos(E/S lista : cad))
var
   lista : act,ant
```

```
inicio
   si cad <> nulo entonces
      si cad↑.info = ' ' entonces
         LBorrar(cad)
      fin_si
      ant ← cad
      act ← cad↑.sig
      mientras act <> nulo hacer
         si act↑.info = ' ' entonces
            LBorrar(ant↑.sig)
         fin_si
         ant ← act
         act ← act↑.sig
      fin_mientras
   fin_si
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
```

2. Almacene en otra lista los signos de puntuación, borrándolas de la lista original.

```
procedimiento BorrarSignos(E/S lista : cad, Signos)
var
   lista : act,ant
inicio
   Sigos ← nulo //Se crea la lista de signos
   si cad <> nulo entonces
      si (cad↑.info = '.') o (cad↑.info = ',') entonces
         LBorrar(cad)
         LInsertar(Signos, cad↑.info)
      fin_si
      ant ← cad
      act ← cad↑.sig
      mientras act <> nulo hacer
         si (cad↑.info = '.') o (cad↑.info = ',') entonces
            LBorrar(ant↑.sig)
            LInsertar(Signos, cad↑.info)
         fin_si
         ant ← act
         act ← act↑.sig
      fin_mientras
   fin_si
fin_procedimiento

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
```

3. Almacene en una lista enlazada las palabras que aparecen en la lista. Las palabras deben estar ordenadas de forma ascendente y las palabras repetidas sólo aparecerán una vez.

```
procedimiento AlmacenarPalabras(E lista : cad ; E/S lista : pal)
var
inicio
   pal ← nulo
   mientras cad <> nulo hacer
      InsertarOrdenado(pal, cad↑.info)
      pal ← pal↑.sig
    fin_mientras
fin_procedimiento

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

   //Sólo hay que incluir el elemento si el elemento no está
   si no encontrado entonces
      //Si hay que insertarlo al comienzo de la lista
      si c = act entonces
          LInsertar(c,e)
      si_no
          LInsertar(ant↑.sig,e)
      fin_si
   fin_si
fin_procedimiento
```

4. Liste las palabras en orden inverso.

```
procedimiento ListadoInverso(E lista : c)
inicio
   si c <> nulo entones
      ListadoInverso(c↑.sig)
      escribir(c↑.info)
   fin_si
fin_procedimiento
```

Puntuación: 3,5 puntos