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	2002
ESPECIALIDAD	Común	CURSO	1º
TURNO	Mañana	CURSO ACADÉMICO	2002/2003
CARÁCTER	Cuatrimestral	PROGRAMA	Ingeniería Superior en Informática
DURACIÓN APROXIMADA			

Pregunta teórica

Colas

Conteste las siguientes preguntas:

1. ¿Qué caracteriza una cola respecto al resto de estructuras lineales de datos?
2. ¿Qué formas de implementar colas conoce? Explique y haga una representación gráfica de cada una de ellas.

Capítulo 12 del libro de texto

Utilizando las operaciones primitivas de colas, realice un algoritmo que permita eliminar las letras mayúsculas de una cola que contiene n caracteres (no es necesario que implemente las operaciones primitivas).

```
procedimiento BorrarMayusculas(E/S cola : c ; E entero :n)
var
   TipoElemento : e
   entero : i
inicio
   desde i ← 1 hasta n hacer
      Primero(c,e)
      CBorrar(c)
      si (e < 'A') o (e > 'Z') entonces
         CInsertar(c,e)
      fin_si
   fin_si
fin_procedimiento
```

Puntuación: 1,5 puntos

Pregunta práctica

Se desean implementar cadenas utilizando una lista enlazada. Cada elemento de la lista será un carácter de la cadena. Codifique los métodos adecuados (sin utilizar el tipo de dato estándar) para:

1. Leer una cadena carácter a carácter hasta que el usuario pulse la tecla Intro.

Declaraciones de tipos de datos

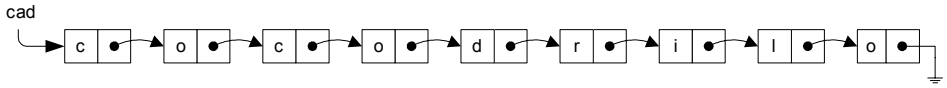
```
tipos
   carácter = TipoElemento //El tipo de elementos del conjunto
   puntero_a nodo = lista //El conjunto sería una lista enlazada
   registro = nodo
      TipoElemento : info
      lista : sig
    fin_registro

procedimiento LeerCadena(E/S : lista : c)
var
   carácter : car
inicio
   leer(car)
   si car = #13 entonces
      crearLista(c)
```

```
   si_no
      LeerCadena(c)
      LInsertar(c,car)
   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

procedimiento CrearLista(E/S lista : l)
inicio
   l ← nulo
fin_procedimiento
```

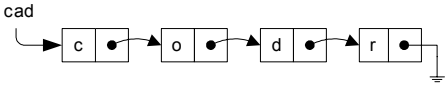
2. Escribir una cadena.

```
procedimiento EscribirCadena(e lista :c)
var
   carácter : car
inicio
   mientras no EsListaVacia(c) hacer
     LPrimero(c,car)
     LSiguiente(c,c)
     escribir(car)
   fin_mientras
fin_procedimiento

lógico: función EsListaVacía(E lista : l)
inicio
   devolver(l = nulo)
fin_función

procedimiento LPrimero(E lista : l; E/S TipoElemento : e)
inicio
   si l = nulo entonces
      // Error, la listaestá vacía
   si_no
      e ← l↑.info
   fin_si
fin_procedimiento

procedimiento LSiguiente(E lista : l; E/S lista  : sig)
inicio
   si l = nulo entonces
      // Error, la listaestá vacía
   si_no
      sig ← l↑.sig
   fin_si
fin_procedimiento
```

3. Realizar un procedimiento `subcadena` que devuelva una subcadena de una cadena principal a partir de una posición el número de caracteres indicado.
   Por ejemplo, si la cadena `cad` es



con la llamada `subcadena(cad,3,4,scad)`, `scad` sería:




```
procedimiento Subcadena(e lista : c ; e entero : ini,fin; e/s lista :sc);
var
  i : entero
 carácter: car
inicio
  crearLista(sc)
  //Mover hasta el caracter ini - 1
  i ← 1
  mientras (i < ini) y no EsListaVacia(c) hacer
       i := i ← 1
       LSiguiente(c,c)
  Fin_mientras
  //coger hasta el caracter fin
  i ← 1
  mientras no EsListaVacia(c) y (i <= fin) hacer
        LPrimero(c,car)
        LInsertar(sc,car)
        i ← i + 1
        LSiguiente(c,c)
  Fin_mientras
  InvertirLista(sc)
fin_procedimiento

procedimiento InvertirLista(E/S lista : l)
var
   lista : act,sig,ant
inicio
   si l <> nulo entonces
      act ← l↑.sig
      ant ← l
      l↑.sig ← nulo
      mientras act <> nulo hacer
         sig ← act↑.sig
         act↑.sig ←ant
         ant ← act
         act ← sig
      fin_mientras
      l ← ant
   fin_si
fin_procdimiento
```

4. Realizar un procedimiento `BorrarCarácter(cad,car)` que elimine todas las apariciones del carácter `car` dentro de la cadena `cad`.

```
procedimiento BorrarCarácter(E/S lista : cad; E carácter : car)
var
   lista : act,ant
inicio
   si cad <> nulo entonces
      si cad↑.info = car entonces
         LBorrar(cad)
      fin_si
```

```
            ant ← cad
            act ← cad↑.sig
            mientras act <> nulo hacer
               si act↑.info = car 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
```

Puntuación: 3,5 puntos