

1. (a) Dado el siguiente arreglo, muestre cómo queda el mismo luego de cada modificación que realiza el algoritmo **insertion sort**.

[3, 17, 2, 10, 5, 8, 3]

- (b) Escriba un algoritmo que, dado un arreglo y un elemento e que no pertenece al mismo, ubique los menores a e al comienzo del arreglo y devuelva la cantidad de elementos menores a e . Utilice para ello el siguiente encabezado:

proc p (in/out a: array[1..n] of Int, in e: Int, out k: Nat)

2. Dado el siguiente procedimiento

```
proc p (in/out l: list)
  var a, b: pointer to node
  a := l
  while a ≠ null do
    b := a → next
    if b ≠ null then
      a → next := b → next
      free(b)
    fi
    a := a → next
  od
end proc
```

donde los tipos **node** y **list** se definen como sigue

```
type node = tuple
  value: elem
  next: pointer to node
end
type list = pointer to node
```

- (a) Explique qué hace el procedimiento p.
- (b) ¿Cuál es el orden del procedimiento p? Justificá la respuesta.
- (c) Si se llama al procedimiento p con una lista que tiene los valores 1, 2, 3, 4, 5, 6 y 7, en ese orden, ¿qué valores tendrá la lista luego de la llamada?

3. Para el siguiente algoritmo, plantee la recurrencia en función de n y calcule el orden contando la cantidad de asignaciones a la variable t :

```
fun f (n: nat) ret t: nat
  if n ≤ 1 then t := 1
  else
    t := f(n/2)
    t := t + f(n/2)
    for i := 1 to n do
      for j := 1 to n do
        t := t + 1
      od
    od
  fi
end fun
```

4. Un **multiconjunto** M con elementos en S , es un subconjunto de S donde cada elemento tiene asociado un número natural que indica cuántas veces el elemento ocurre. En otras palabras, un multiconjunto puede pensarse como un conjunto en el cual los elementos pueden estar incluidos más de una vez, indicando cuántas veces lo hace.

(a) Especificar el TAD **Multiset of T** de multiconjuntos con elementos de tipo T , incluyendo constructores para crear un multiconjunto vacío y otro para agregar un elemento a un multiconjunto. Y operaciones para saber si un multiconjunto es vacío o no, para saber si un elemento dado existe en el multiconjunto, para obtener cuántas veces un elemento dado ocurre en un multiconjunto, y para eliminar una ocurrencia de un elemento en un multiconjunto.

(b) Implementar el TAD **Multiset of T** especificado en el punto anterior utilizando como representación interna una lista donde cada elemento es un par consistente de un elemento de tipo T y un número natural que indica la cantidad de ocurrencias.

(c) Implementar una función que reciba un arreglo de N naturales y devuelva el multiconjunto que contiene todos los elementos pares que ocurren en el arreglo. Todo tipo de datos utilizado en la función debe utilizarse de manera **abstracta**.

004 → 40

HOJA N° 1

FECHA

TOMÁS ARIAN BARRERA

EJERCICIO 1:

q) PASO A PASO DE INSERTION SORT

$A = [3, 17, 2, 10, 5, 8, 3] \rightarrow [3, 17, 2, 10, 5, 8, 3]$

$i=1$ EL SEGMENTO $A[1..1]$

ESTA ORDENADO, NO HAGO CAMBIOS

PORTE ORDENADA =

$i=2$

COMO $17 > 3$, EL SEGMENTO $A[1..2]$

ESTA ORDENADO, NO HAGO CAMBIOS

$\rightarrow [3, 17, 2, 10, 5, 8, 3] \rightarrow [3, 2, 17, 10, 5, 8, 3]$

COMO $2 < 17$, EL PROCEDIMIENTO INSERT REALIZA UN SWAP ENTRE ELLOS. LUEGO COMO $2 < 3$, EL PROCEDIMIENTO INSERT REALIZA OTRO SWAP Y FINALIZA.

$\rightarrow [2, 3, 17, 10, 5, 8, 3] \rightarrow [2, 3, 17, 10, 5, 8, 3]$

$i=4$

COMO $10 < 17$, EL PROC INSERT LOS INTERCAMBIA

$\rightarrow [2, 3, 10, 17, 5, 8, 3] \rightarrow [2, 3, 10, 17, 5, 8, 3]$

$i=5$

COMO $5 < 17$, INSERT LOS INTERCAMBIA

$\rightarrow [2, 3, 10, 5, 17, 8, 3] \rightarrow [2, 3, 5, 10, 17, 8, 3]$

COMO $5 < 10$, INSERT LOS INTERCAMBIA

$\rightarrow [2, 3, 5, 10, 17, 8, 3] \rightarrow [2, 3, 5, 10, 8, 17, 3] \rightarrow [2, 3, 5, 8, 10, 17, 3]$

$i=6$ COMO $8 < 17$ Y $8 < 10$, INSERT LOS INTERCAMBIA EN ORDEN

$\rightarrow [2, 3, 5, 8, 10, 17, 3] \rightarrow [2, 3, 5, 8, 10, 3, 17] \rightarrow [2, 3, 5, 8, 3, 10, 17]$

$i=7$ COMO $3 < 17$, $3 < 10$, $3 < 8$ Y $3 < 5$ ENTONCES EL ELEMENTO ES "SWAPPEADO"

$\rightarrow [2, 3, 5, 3, 8, 10, 17] \rightarrow [2, 3, 3, 5, 8, 10, 17]$

HACIA LA IZQUIERDA 4 VECES.

FIN (ARRAY ORDENADO)

Ejercicio 3:

```

FUN F(n: nat) ret t: nat
IF n ≤ 1 then t := 1
ELSE
  T := F(n/2)
  T := T + F(n/2)
  FOR i := 1 TO n DO
    FOR j := 1 TO n DO
      T := T + 1
    OD
  OD
FI
END FUN
  
```

FUNCION DE RECURSIVIDAD

$$T(n) = \begin{cases} 1 & \text{Si } n \leq 1 \\ 2 \cdot T(n/2) + n^2 + 2 & \text{EN OTRO CASO} \end{cases}$$

$a = 2$ Como $a < b^K$ ($2 < 2^2$)
 $b = 2$ ENTONCES EL PROCEDIMIENTO TIENE
 $K = 2$ ORDEN n^K , ES DECIR, n^2 .

ESTE RESULTADO SALE DE QUE EL ORDEN DE LA FUNCION DE RECURSIVIDAD ESTA DADO POR:

$$\begin{cases} n^{\log_b(a)} & \text{Si } a > b^K \\ n^K \log(n) & \text{Si } a = b^K \\ n^K & \text{Si } a < b^K \end{cases}$$

ALVARO BLAZO

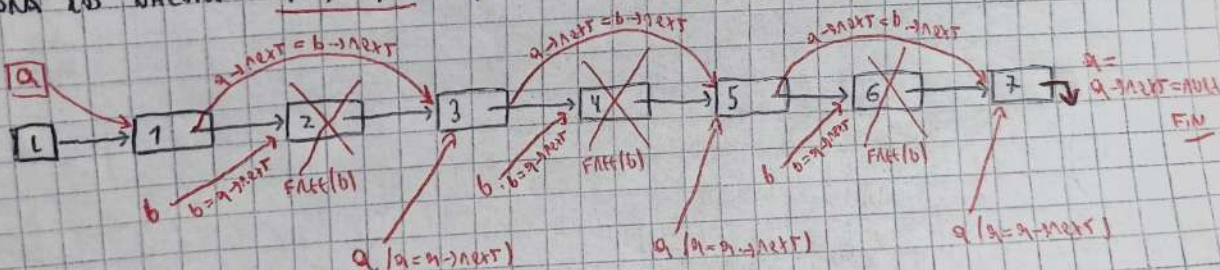
100%

EJERCICIO 2:

a) El procedimiento P elimina los nodos en posiciones impares de la lista (si el primer elemento está en la posición 0).

b) El orden del procedimiento P es n , donde n es la longitud de la lista. Esto es debido a que el ciclo while recorre una única vez a todos los elementos de la lista, y no hay recursión sobre P . Esto significa que la máxima cantidad de asignaciones no será más que un múltiplo de n , y por lo tanto el procedimiento es de orden n . (también se utilizarían funciones de orden $\geq n$)

c) Si se llama a P con la lista 1, 2, 3, 4, 5, 6, 7, luego de su ejecución la lista tendrá los valores 1, 3, 5 y 7.

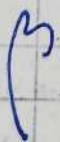


(escribí yo con rojo, no es que está mal)

EXPLICACIÓN
GRÁFICA

b)

```
PROC P (in/out a: ARRAY[1..N] OF Int, in e: Int, out k: Nat)
  var lft: Nat
  lft := 1
  FOR i := 1 TO N DO
    IF (a[i] < e) THEN
      SWAP(a, i, lft)
      lft := lft + 1
    FI
  OD
  k := lft - 1
END PROC
```



9) SPEC MULTISSET OF T where

CONSTRUCTORS

FUN empty-multiset() ret M: MULTISSET OF T
 { CREA UN MULTICONTUNTO VACÍO }

PROC addelem (in/out M: MULTISSET OF T, in e: T)
 { AÑADE EL ELEMENTO e AL MULTICONTUNTO M }

OPERATIONS

FUN is-empty-multiset (M: MULTISSET OF T) ret b: bool
 { DEVUELVE TRUE SI EL MULTICONTUNTO M ESTÁ VACÍO }

FUN elem-exists (M: MULTISSET OF T, e: T) ret b: bool
 { DEVUELVE TRUE SI EL ELEMENTO e ESTÁ EN M }

FUN elem-amount (M: MULTISSET OF T, e: T) ret a: nat
 { DEVUELVE LA CANTIDAD "a" DE VECES QUE e ESTÁ EN M }

PROC delete_one (in/out M: MULTISSET OF T, e: T)
 { ELIMINA UNA OCURRENCIA DE e EN M }

PROC destroyM (in/out M: MULTISSET OF T)
 { DESTROYE EL MULTICONTUNTO M ELIMINANDO TODOS SUS ELEMENTOS }

b) IMPLEMENT MULTISSET OF T where

TYPE MULTISSET OF T = LIST OF (SETS OF T)

TYPE SETS = TUPLE
 OF T
 ELEMENT: T
 AMOUNT: nat
 END TUPLE

FUN empty-multiset() ret M: MULTISSET OF T
 M := empty()
 END FUN

PROC addelem (in/out M: MULTISSET OF T, in e: T)

VAR K: SETS OF T

K.elem := e

IF (elem-exists(M, e)) then

while (head(M).element \neq e) do

addl(head(M))

TAIL(M)

OD

K.AMOUNT := HEAD(M).AMOUNT + 1

TAIL(M)

addl(M, K)

ELSE

K.AMOUNT := 1

addl(M, K)

FI

END PROC

esta eliminado
 la cabeza
 de la lista.

Definición de la función


```

IS-EMPTY-MULTISET (M: MULTISSET OF T) ret b: bool
b := IS-EMPTY(M)
END FUN

```

```

FUN ELEM-EXISTS (M: MULTISSET OF T, e: T) ret b: bool
b := FALSE
var i: nat
i := 0
while (not b and i < length(M)) do
    b := b v (index(M, i).element = e)
    i := i + 1
od
END FUN

```

```

FUN elem-amount (M: MULTISSET OF T, e: T) ret a: nat
IF (not elem-exists(M, e)) then a := 0
ELSE
    var i: nat
    i := 0
    while (index(M, i).element ≠ e) do
        i := i + 1
    od
    a := index(M, i).amount
FI
END FUN

```

```

PROC delete-one (in/out M: MULTISSET OF T, in e: T)
var k: nat
k := elem-amount(M, e)
IF k = 0 THEN skip
ELSE
    while (head(M).element = e) do
        addr(head(M))
        tail(M)
    od
    IF k = 1 THEN tail(M)
    ELSE
        var aux: SET OF T
        aux.element := e
        aux.amount := head(M).amount - 1
        tail(M)
        addl(M, aux)
    FI
FI
END PROC

```

```

PROC destroyM (M: MULTISSET OF T)
destroy(M)
END PROC

```

*Note: (at functions destroy, addl, addr, head, tail, etc) you use conditions like TAD list of T.

ACHAVAL (kntto)

Ejercicio 4.c)

```
FUN EVENS_MULTSET (a: array[1..N] of Nat) ret M: Multiset of Nat
  FOR i:=1 TO N DO
    IF ((a[i] MOD 2) ≠ 0) THEN
      addElem(M, a[i])
    FI
  OD
END FUN
```