Algoritmos y Estructuras de Datos II - 22 de abril de 2019

Primer Parcial

Alumno: .....

Siempre se debe explicar la solución, una respuesta correcta no es suficiente si no viene acompañada de una justificación que demuestre que la misma ha sido comprendida. Las explicaciones deben ser completas. En el ejercicio de implementación, debe utilizarse pseudo-código. **La utilización de código c influirá negativamente.**

1. (a) Dado el siguiente arreglo, muestre cómo queda el mismo luego de cada modificación que realizan los algoritmos **selection sort** e **insertion sort**.

$$[6, 20, 1, 12, 7, 9, 4]$$

   (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. Dada la siguiente función:

```
fun f(n: nat) ret m: nat
    if n ≤ 1 then m:= 2 * n
    else m:= 1
         for i:= n downto 1 do m:= n * m od
         m:= 3 * f(n div 2)
    fi
end fun
```

   (a) ¿Cuántas llamadas recursivas a f(n **div** 2) se realizan durante la ejecución de f(n)?

   (b) Exprese la ecuación de recurrencia en función de la cantidad de asignaciones a la variable m.

   (c) Calcule el orden de asignaciones a la variable m.

3. 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. Justifique.

   (b) ¿Cuál es el orden del procedimiento p? Justifique.

   (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?

4. a) Completar la especificación del TAD Conjunto

```
module TADConjunto (Conjunto, vacíoC, consC, esvacíoC, elemC, inclC, unionC, interC) where

data Conjunto e = Vacío | Cons e (Conjunto e) deriving Show

-- ECUACIONES ENTRE CONSTRUCTORES
-- Cons e (Cons e c) == Cons e c
-- Cons e (Cons f c) == Cons f (Cons e c)

vacíoC = Vacío
consC = Cons

esvacíoC :: Conjunto e → Bool
elemC :: Eq e ⇒ e → Conjunto e → Bool
inclC :: ...     -- inclusión
unionC :: ...    -- unión
interC :: ...    -- intersección

esvacíoC Vacío = True
esvacíoC c = False

e `elemC` Vacío = False
e `elemC` (Cons f c) = e == f || e `elemC` c

Vacío `inclC` d = True
Cons e c `inclC` d = ...

Vacío `unionC` d = ...
Cons e c `unionC` d = ...

Vacío `interC` d = ...
Cons e c `interC` d | e `elemC` d = ...
                    | otherwise = ...

instance Eq e ⇒ Eq (Conjunto e) where
    c == d = c `inclC` d && d `inclC` c
```

b) Completar la especificación del TAD Grafo

```
module TADGrafo (Grafo, vacíoG, aristaG,
        esvacíoG, estávérticeG, estáaristaG,
        vecinosG, subG, haycaminoG) where

import TADConjunto

data Grafo e = Vacío | Arista e e (Grafo e)
             deriving Show

-- ECUACIONES ENTRE CONSTRUCTORES
-- Arista a b (Arista a b g) == ...
-- Arista a b (Arista c d g) == ...

vacíoG = Vacío
aristaG = Arista

esvacíoG :: Grafo e → Bool
estávérticeG :: ... si el vértice está en el grafo
estáaristaG :: ...  si la arista está en el grafo
vecinosG :: ...  conjunto de vecinos del vértice
subG :: ... si uno es subgrafo del otro
haycaminoG :: ... si hay camino de v a w

esvacíoG ... = ...

estávérticeG v Vacío = ...
estávérticeG v (Arista a b g) = ...

estáaristaG v w Vacío = ...
estáaristaG v w (Arista a b g) = ...

vecinosG v Vacío = ...
vecinosG v (Arista a b g) | v == a = ...
                          | otherwise = ...

subG Vacío h = True
subG (Arista a b g) h = ...

haycaminoG v w Vacío = ...
haycaminoG v w (Arista a b g) = ...

instance Eq e ⇒ Eq (Grafo e) where
    ...
```

Tener en cuenta que Arista a b indica la presencia de una arista del vértice a al vértice b. Los vecinos del vértice v son los vértices w tales que hay una arista de v a w. Un camino de v a w consiste de una secuencia de cero o más aristas que permiten ir de v a w en cero o más pasos. Un grafo es subgrafo de otro si todas las aristas del primero están en el segundo.