Apellidos:
Nombre:
Convocatoria:
DNI:

Examen PED junio 2016
Modalidad 0

Normas:

- Tiempo para efectuar el test: **20 minutos**.
- Una pregunta mal contestada elimina una correcta.
- Las soluciones al examen se dejarán en el campus virtual.
- Este test vale 2 puntos (sobre 10).
- **Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
- En la **hoja de contestaciones** el verdadero se corresponderá con la **A**, y el falso con la **B**.

	V	F		
En la inserción de un elemento en un árbol 2-3, la altura del árbol resultado siempre crece (con respecto al árbol original) cuando la raíz del árbol original es un 3-nodo.	❑	❑	1	F
En la inserción de un elemento en un árbol 2-3-4, la altura del árbol resultado siempre crece (con respecto al árbol original) cuando la raíz del árbol original es un 4-nodo.	❑	❑	2	V
En el algoritmo de borrado de un elemento en un árbol 2-3-4, siempre que el nodo "q" sea 2-nodo hay que hacer reestructuraciones.	❑	❑	3	V
La complejidad temporal de la operación desapilar (vista en clase) utilizando vectores (con un índice que indica la cima de la pila) o utilizando listas enlazadas es la misma.	❑	❑	4	V
La semántica de la operación quita_hojas que actúa sobre un árbol binario y devuelve el árbol binario original sin sus hojas es la siguiente: VAR i, d: arbin; x: item; quita_hojas(crea_arbin( )) = crea_arbin( ) quita_hojas(enraizar(crea_arbin(), x, crea_arbin()) = enraizar(crea_arbin(), x, crea_arbin() quita_hojas(enraizar(i, x, d)) = enraizar(quita_hojas(i), x, quita_hojas(d))	❑	❑	5	F
Todo árbol mínimo es un árbol binario de búsqueda	❑	❑	6	F
El grado de los árboles AVL puede ser +1, 0 ó -1.	❑	❑	7	F
Todo árbol binario de búsqueda es un árbol 2-3.	❑	❑	8	F
En un árbol 2-3-4 el máximo número elementos del nivel N es $3*2^{2N-2}$	❑	❑	9	V
La especificación algebraica de la siguiente operación indica que se devolverá el número de elementos del conjunto multiplicado por 3 (C: Conjunto; x: Ítem): Operación(Crear) $\Leftrightarrow$ 0 Operación (Insertar(C, x)) $\Leftrightarrow$ 3 + Operación(C)	❑	❑	10	V
En el TAD Diccionario con dispersión cerrada, con función de redispersión "hi(x)=(H(x) + k(x)*i) MOD B", con B=6 se puede dar la situación de que en una búsqueda no se acceda a todas las posiciones de la tabla.	❑	❑	11	V
En un Hash cerrado con factor de carga α, se cumple que $0 \leq \alpha \leq 1$	❑	❑	12	V
En un montículo doble, un elemento "j" del montículo máximo es el simétrico de un único elemento "i" del montículo mínimo.	❑	❑	13	F
Un multigrafo es un grafo que no tiene ninguna restricción: pueden existir arcos reflexivos y múltiples ocurrencias del mismo arco.	❑	❑	14	V
Sea G=(V,A) un grafo dirigido. Diremos que G"=(V",A") es un árbol extendido de G $\Leftrightarrow$ V"=V, A"$\subset$ A, $\forall$ v$\in$V" $\Rightarrow$ gradoE(v) $\leq$ 1	❑	❑	15	V

Examen PED junio 2016

Normas:
- Tiempo para efectuar el examen: **2 horas**
- En la cabecera de cada hoja **Y EN ESTE ORDEN** hay que poner: ***APELLIDOS, NOMBRE***.
- Cada pregunta se escribirá en hojas diferentes.
- Las soluciones al examen se dejarán en el campus virtual.
- Se puede escribir el examen con lápiz, siempre que sea legible
- **Cada pregunta vale 2 puntos (sobre 10).**
- **Las fechas de "Publicación de notas" y "Revisión del examen teórico" se publicarán en el Campus Virtual.**

1. Dado el **grafo no dirigido** representado por la lista de adyacencia que se muestra a continuación:

1 → 2→ 4
2 → 9
3 → 1→ 7
5 → 6 → 3→ 1
6 → 1
7 → 1 → 5
8 → 7
9 → 1
10 → 11
11 → 13
12 → 14 → 13
13 → 10
14 → 10 → 11

a) Obtener DFS(11), el árbol extendido en profundidad partiendo del vértice 11 y la clasificación de las aristas.

b) Obtener BFS(11), el árbol extendido en anchura partiendo del vértice 11 y la clasificación de las aristas.

Nota: La lista de adyacencia de cada vértice se recorre de mayor a menor vértice para todos los casos del ejercicio. Las listas están desordenadas.

c) Utilizando exclusivamente las operaciones constructoras generadoras del tipo grafo, definid la semántica de la operación examen que se aplica sobre un grafo dirigido ponderado cuyos arcos están etiquetados con números naturales (es decir, el peso de los arcos son números naturales) y devuelve el número de arcos cuyo peso es igual a uno especificado. La sintaxis de la operación 'examen' es la siguiente:

examen: grafo, natural_peso → *natural*

2. a) Definir la sintaxis y la semántica de la función *Camino* que recibe como parámetros una lista y un árbol binario y devuelve un booleano. Esta función devolverá TRUE si la lista de elementos pasada como parámetro forma un camino válido en el árbol binario pasado como parámetro. En otro caso devolverá FALSE. Nota: Los elementos de la lista y del árbol binario son naturales. Se podrán utilizar todas las operaciones definidas en clase para los TADs mencionados.

b) Insertar en una tabla de dispersión cerrada de tamaño B=7, con función de dispersión H(x) = x MOD B y con estrategia de redispersión segunda función hash, los siguientes elementos: 2341, 4123, 83, 911, 5211, 3411, 52.

Indica el número total de intentos necesarios para almacenar todos los elementos.

Nota: La información utilizada para almacenar cada elemento en la tabla HASH será la suma de las cifras de cada elemento. Por ejemplo, para almacenar el elemento 2341 tendríamos 2+3+4+1=10, por tanto, H(2341)=10 MOD 7=3. El elemento 2341 iría en la posición 3 de la tabla HASH.

3. Dada la siguiente función que ordena un vector:

```
ORD  (a:vector[natural];  n: natural)
var i,j: entero;  x:natural  fvar
comienzo
      para  i:=2  hasta  n  hacer
            x:=a[i];  j:=i-1
            mientras   (j>0)  AND  (a[j]>x)
                  a[j+1]:=a[j]
                  j:=j-1
            fmientras
            a[j+1]:=x
      fpara
fin
```

a) Indicar razonadamente la complejidad temporal en su caso **mejor y peor**.

b) Aplicar **detalladamente** dicho algoritmo para ordenar el siguiente vector: [40, 3, 5, 9, 12, 30, 2, 35, 1].

c) Conseguir **la misma** ordenación del vector anterior mediante el algoritmo Heapsort visto en clase. NOTA: es obligatorio explicar cómo se realizan los intercambios utilizando únicamente el vector.

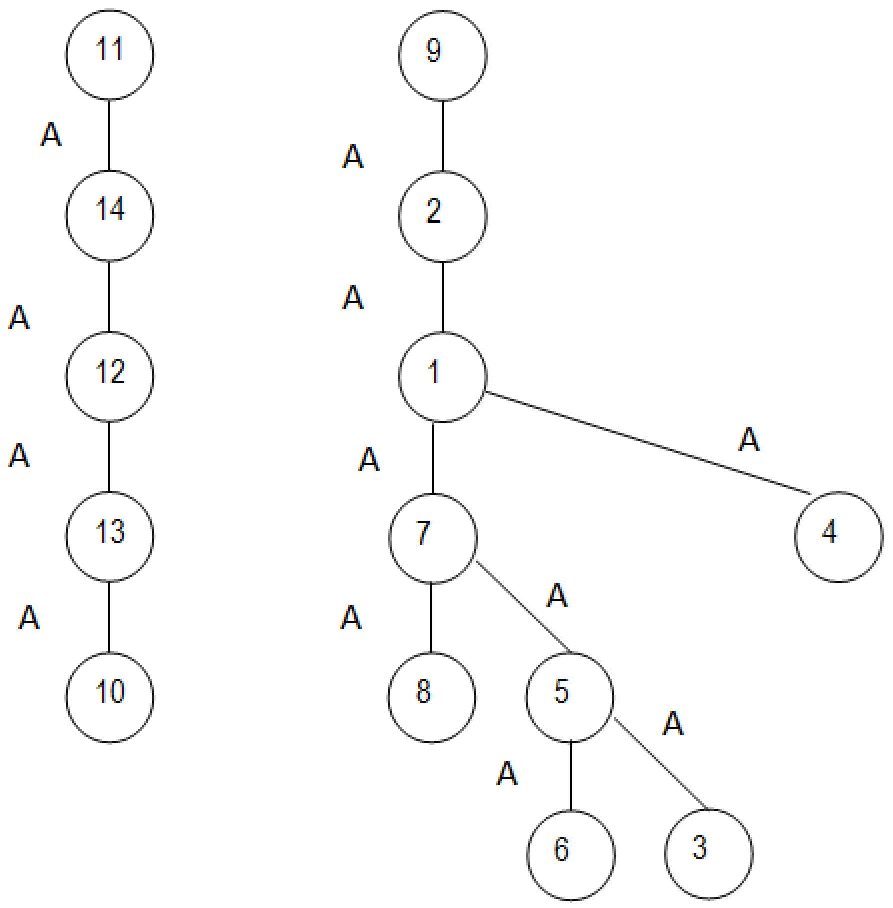
d) Calcular razonadamente la complejidad del Heapsort en su caso **mejor y peor**.

Examen PED junio 2016. Soluciones

1.

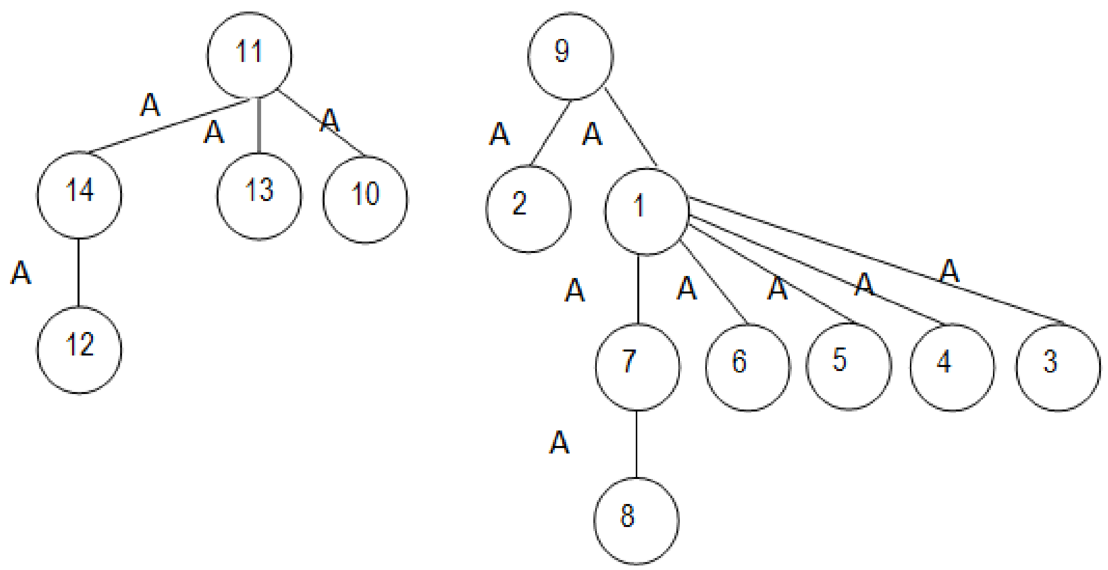
a) DFS(11)=11,14,12,13,10. Se continúa por DFS(9)=9,2,1,7,8,5,6,3,4

Árbol extendido en profundidad. Las aristas marcadas son de árbol (A), el resto son de retroceso



b) BFS(11)=11,14,13,10,12. Se continúa por BFS(9)=9,2,1,7,6,5,4,3,8

Árbol extendido en anchura. Las aristas marcadas son de árbol (A), el resto son de retroceso



c) Var G: grafo; x,y: vértice; p,q: natural;

examen(crear_grafo(),q)=0
examen(InsertarArista(G,x,y,p),q)=
si (q ==p) entonces 1 + examen(G,q)
    si no examen(G,q)

2.

a)

La sintaxis de la función sería: Camino(Lista, AB) → bool
VAR x,y: Natural; L:Lista; i,d: AB

```
Camino(crear(), crear()) = TRUE
Camino(crear(), enraizar(i, x, d) = TRUE
Camino(inscabeza(L,x), crear()) = FALSE
Camino(inscabeza(L,x), enraizar(i, y, d)) =
   Si (x!=y) entonces Camino(inscabeza(L, x), i) OR Camino(inscabeza(L, x), d) Fsi
      Si (x==y) entonces
            Si (es_vacia(L) entonces TRUE
            Sino
         Si (obtener(L, primera(L)) == raiz(i))  entonces Camino(L, i)
               Sino
                  Si (obtener(L, primera(L)) == raiz(d)  entonces Camino(L, d)
                  Sino FALSE
                  Fsi
               Fsi
            Fsi
      Fsi
```

b)

$K(x) = (x \text{ MOD } (B-1)) + 1$
$h_i(x) = (h_{i-1}(x) + k(x)) \text{ MOD } B$

H(2341) = 10 MOD 7 = 3

H(4123) = 10 MOD 7 = 3
k(4123) = (10 MOD 6) +1 = 5
h_1(4123) = (3+5) MOD 7 = 1

H(83) = 11 MOD 7 = 4

H(911) = 11 MOD 7 = 4
k(911) = (11 MOD 6) +1 = 6
h_1(911) = (4 + 6) MOD 7 = 3
h_2(911) = (3 + 6) MOD 7 = 2

H(5211) = 9 MOD 7 = 2
k(5211) = (9 MOD 6) +1 = 4
h_1(5211) = (2 + 4) MOD 7 = 6

H(3411) = 9 MOD 7 = 2
k(3411) = (9 MOD 6) +1 = 4
h_1(3411) = (2 + 4) MOD 7 = 6
h_2(3411) = (6 +4) MOD 7 = 3
h_3(3411) = (3 + 4) MOD 7 = 0

H(52) = 7 MOD 7 = 0
k(52) = (7 MOD 6) +1 = 2
h_1(52) = (0 + 2) MOD 7 = 2
h_2(52) = (2 + 2) MOD 7 = 4
h_3(52) = (4 + 2) MOD 7 = 6
h_4(52) = (6 + 2) MOD 7 = 1
h_5(52) = (1 + 2) MOD 7 = 3

0	3411
1	4123
2	911
3	2341
4	83
5	52
6	5211

Número de intentos totales = 20

$h_6(52) = (3 + 2)$ MOD $7 = 5$

3.

a) Caso mejor $\Omega(n)$: en el mientras no entra nunca. El último elemento de destino es siempre el menor y como destino ya está ordenado → vector ordenado ascendentemente

$$\sum_{i=2..n} 1 = n-1 \in \Omega(n)$$

Caso peor $O(n^2)$: todos son mayores en destino. Hay que desplazarlos todos →vector ordenado inverso.

$$\sum_{i=2..n} (1+ \sum_{j=1..i-1} 1) = \sum_{i=2..n} (1+i-1) \in O(n^2)$$

c) Puesto que se realiza una ordenación de menor a mayor, habrá de utilizarse un montículo máximo.

d) $\Omega(n \log_2 n) = O(n \log_2 n)$ Ya que aunque esté ordenado previamente, en la creación/borrado del montículo ha de hacer n operaciones, cada una con un coste de $\log_2 n$ (la altura del montículo).