

# Algoritmos y Estructuras de Datos.

## Examen Final y Recuperatorio. [17 de Julio de 2003]

**Ej. 1.- [Primitivas (Recup=10 puntos, Final=20 puntos)]** En todos los casos *escribir todos los tipos, definiciones, funciones y procedimientos auxiliares necesarios.*

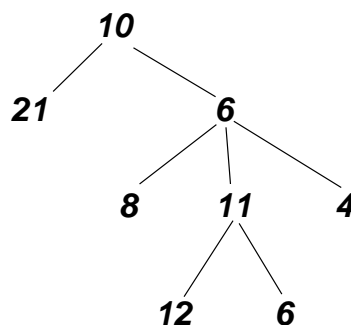
- [Alumnos que recuperan 1er parcial]** Escribir las funciones del TAD COLA implementadas por **arreglo circular**, a saber: ANULA(C), PONE\_EN\_COLA(x,C), QUITA\_DE\_COLA(C), VACIA(C), y FRENTE\_DE\_COLA(C).
- [FINAL+Alumnos que recuperan 2do parcial]** Escribir las funciones del TAD ARBOL BINARIO con celdas enlazadas por **punteros ó cursores** a saber: PADRE(n,A), HIJO\_IZQ(n,A), HIJO\_DER(n,A), ETIQUETA(n,A), CREA2(v,A1,A2) y ANULA(A).
- [Alumnos que recuperan 3er parcial]** Escribir las funciones del TAD CONJUNTO implementado mediante **vectores de bits** a saber: ANULA, UNION, INTERSECCION, MIEMBRO, MIN, INSERTA y SUPRIME.

**Ej. 2.- [TODO: Ejercicios de programación]**

- [Expande (Recup=20 puntos, Final=35 puntos)]** Escribir un procedimiento **procedure EXPANDE(var L:lista);** que inserta una sucesión de números enteros a partir de la posición que esta siendo visitada. La cantidad de números a insertar debe ser igual al número entero que esta en esa posición. La sucesión de enteros debe comenzar con el entero con el cual se va a expandir y debe ser creciente. Por ejemplo, si inicialmente  $L=\{1, 2, 3, 4\}$ , después de hacer EXPANDE(L) debe quedar  $L=\{1, 2, 2, 3, 4, 3, 4, 5, 6, 4, 5, 6, 7, 8\}$ . Utilizar las primitivas del TAD LISTA: INSERTA(x,p,L), RECUPERA(p,L), SUPRIME(p,L), SIGUIENTE(p,L), ANULA(L), PRIMERO(L), y FIN(L).
- [Verifica ABB (Recup=30 puntos, Final=45 puntos)]** Escribir una función **VERIFICA\_ABB(n: nodo; A: arbol): boolean** que verifica si el subárbol de un nodo n verifica la condición de *árbol binario de búsqueda*.

**Ej. 3.- [LIBRES y RECUPERATORIO. Ejercicios operativos]**

- [Crea (Recup=10 puntos, Final=25 puntos)]** Escribir una función **function CREA\_ARBOL(A:arbol): nodo** que, usando las funciones del TAD ARBOL ORDENADO ORIENTADO CREA0(v), CREA1(v, A1), ... CREA<sub>n</sub>(v, A1, A2, ..., An) crea el siguiente árbol

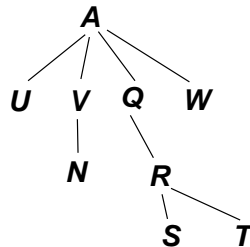


Apellido y Nombre: \_\_\_\_\_

Carrera: \_\_\_\_\_ DNI: \_\_\_\_\_

[Llenar con letra mayúscula de imprenta GRANDE]

- (b) [Particionar (Recup=10 puntos, Final=25 puntos)] Considerando el árbol de la figura, decir cuál son los nodos DESCENDIENTES(Q), ANTECESORES(Q), IZQUIERDA(Q) y DERECHA(Q).



- (c) [Heap-sort (Recup=10 puntos, Final=25 puntos)] Dados los enteros {3, 5, 8, 1, 2, 6, 3, 4, 7} ordenarlos por el método de “montículos” (“heap-sort”). Mostrar el montículo (minimal) antes y después de cada inserción/supresión.

Ej. 4.- [LIBRES(total=25 pto) y RECUPERATORIO(total=10 pto)] Responder según el sistema “multiple choice”, es decir marcar con una cruz el casillero apropiado. **Atención:** Algunas respuestas son intencionalmente “descabelladas” y tienen puntajes **negativos!!**

- (a) ¿Cómo es el tiempo de ejecución para intercalar dos listas clasificadas de  $n$  elementos?

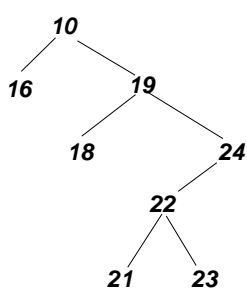
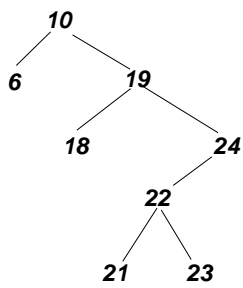
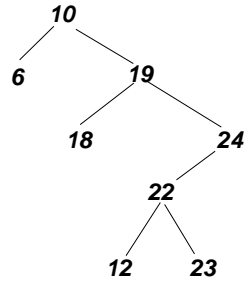
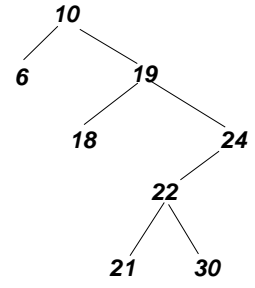
☐ ...  $O(1)$

☐ ...  $O(n)$

☐ ...  $O(n^2)$

☐ ...  $O(\log n)$

- (b) ¿Cuál de los siguientes árboles es un árbol binario de búsqueda?

☐☐☐☐

- (c) El tiempo de ejecución para el algoritmo de clasificación rápida (quick-sort) en el peor caso es:

☐ ...  $O(n \log n)$

☐ ...  $O(1)$

☐ ...  $O(n^2)$

☐ ...  $O(n)$

- (d) El tiempo de ejecución para la función SUPRIME\_MIN en el TAD COLA DE PRIORIDAD implementado por montículos es:

☐ ...  $O(n)$

☐ ...  $O(n \log n)$

☐ ...  $O(\log n)$

☐ ...  $O(1)$