Apellido y Nombre: _____

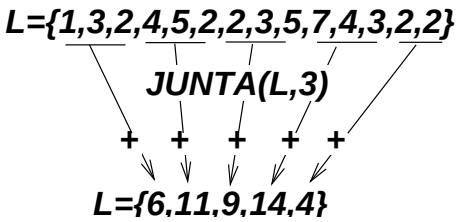
Carrera: _____ DNI: _____
[Llenar con letra mayúscula de imprenta GRANDE]

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática

Algoritmos y Estructuras de Datos.
Examen Final. [1 de Agosto de 2002]

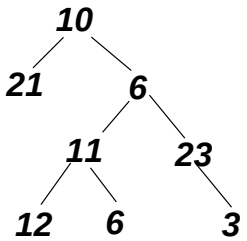
Ej. 1.- Escribir las funciones primitivas del `TAD CONJUNTO` implementado *listas enlazadas clasificadas*. Es decir, implementar en Pascal los siguientes procedimientos/funciones listados abajo. Incluir todas las definiciones de tipo necesarias. (a) `ANULA(A)`, (b) `UNION(A,B,C)`, (c) `INTERSECCION(A,B,C)`, (d) `DIFERENCIA(A,B,C)`, (e) `MIEMBRO(x,A)`, (f) `MIN(A)`, (g) `INSERTA(x,A)` y (h) `SUPRIME(x,A)`.

Ej. 2.- Escribir un procedimiento `procedure JUNTA(var L: lista; n:integer);` que dada una lista `L`, agrupa de a `m` elementos dejando su suma (ver figura). Usar las siguientes primitivas del `TAD LISTA`: `INSERTA(x,p,L)`, `RECUPERA(p,L)`, `SUPRIME(p,L)`, `SIGUIENTE(p,L)`, `ANULA(L)`, `PRIMERO(L)`, y `FIN(L)`. **N**o usar ninguna estructura auxiliar. Prestar atención a no usar posiciones inválidas después de una supresión. El algoritmo debe tener un tiempo de ejecución $O(n)$, donde n es el número de elementos en la lista original.



Ej. 3.- Uso básico de TAD's:

(a) Escribir una función `function MAXIMO_PAR(n:nodo; A:arbol):integer;` que retorna el máximo de las etiquetas **pares** de un árbol binario. En el caso del árbol de la figura debe retornar 12. Usar las primitivas del `TAD ARBOL BINARIO`: `HIJO_IZQUIERDO(n,A)`, `HIJO_DERECHO(n,A)` y `ETIQUETA(n,A)`,



(b) Escribir un procedimiento `procedure ELIMINA_VALOR(var C:cola; n: integer);` que elimina todos las ocurrencias del valor `n` en la cola `C`. Por ejemplo, si `C = {1,3,5,4,2,3,7,3,5}`, después de `ELIMINA_VALOR(C,3)` debe quedar `C = {1,5,4,2,7,5}`. *Sugerencia: Usar una estructura auxiliar lista o cola.* Utilizar las primitivas del **TAD COLA:** `ANULA(C)`, `PONE_EN_COLA(x,C)`, `QUITA_DE_COLA(C)`, `VACIA(C)`, y `FRENTE_DE_COLA(C)`. El algoritmo debe tener un tiempo de ejecución $O(n)$, donde n es el número de elementos en la cola original.

Ej. 4.- [LIBRES] Ejercicios operativos:

(a) **Árboles:** Dibujar el árbol ordenado orientado cuyos nodos, listados en orden previo y posterior son
- `ORD_PRE` $=\{C, D, E, R, S, Q, A, B\}$,
- `ORD_POST` $=\{D, Q, R, A, B, S, E, C\}$.

(b) **Árboles de Huffman:** Dados los caracteres siguientes con sus correspondientes probabilidades, contruir el código binario y encodar la palabra `CORRALITO`
$P(C) = 0.2, P(O) = 0.2, P(R) = 0.2, P(A) = 0.1, P(L) = 0.1, P(I) = 0.1, P(T) = 0.1$
Calcular la longitud promedio del código obtenido.

Ej. 5.- [LIBRES] Preguntas: [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) El tiempo de ejecución para el algoritmo de clasificación rápida (*"quicksort"*) **en el peor caso** es
- $O(\log n)$
- $O(n)$
- $O(n^2)$
- $O(n \log n)$

(b) El tiempo de ejecución para el algoritmo de clasificación por montículos es
- $O(n \log n)$ en el mejor caso, $O(n^2)$ en promedio
- siempre $O(n \log n)$
- siempre $O(n^2)$
- a veces $O(n^2)$

(c) Sea un tabla de dispersión abierta con B cubetas y n elementos. Asumiendo que la función de dispersión es lo suficientemente buena como para distribuir los elementos en forma uniforme entre las cubetas, el costo medio de inserción de un nuevo elemento es
- $O(n^2/B)$
- $O((n/B)^2)$
- $O(n + B)$
- $O(n/B)$