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. [30 de Diciembre de 2003]

Ej. 1.- [**Primitivas (20 puntos)**] Escribir las funciones del `TAD LISTA` con celdas simplemente enlazadas por punteros **ó** cursores. `INSERTA(x,p,L)`, `LOCALIZA(x,L)`, `RECUPERA(p,L)`, `SUPRIME(p,L)`, `SIGUIENTE(p,L)`, `ANULA(L)`, `PRIMERO(L)`, y `FIN(L)`. *Escribir todos los tipos, definiciones, funciones y procedimientos auxiliares necesarios.*

Ej. 2.- [**Ejercicios de programación (total 80 puntos)**]

a) [**check-sum (35 puntos)**] Dadas dos listas de enteros `L1` y `L2` ecribir una función `function CHECK_SUM(L1,L2: lista): boolean;` que retorna verdadero si los elementos de `L1` pueden agruparse sumando de manera de obtener los elementos de `L2` *sin alterar el orden de los elementos*. Por ejemplo, en el caso de la figura `CHECK_SUM(L1,L2)` debe retornar verdadero ya que los agrupamientos mostrados reducen la lista `L1` a la `L2`.

L1={ 1,2,3, 3,2,5, 4,1, 6, 8,3}

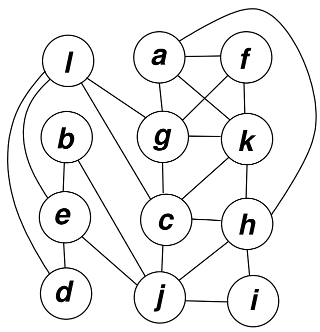
L2={ 6, 10, 5, 6, 11}

b) [**saca-par (20 puntos)**] Escribir un procedimiento `procedure SACA_PAR(var L:lista; C:cola);` que apendiza a la lista `L` todos los elementos de `C` que son pares, los cuales a su vez deben ser removidos de `C`. Por ejemplo, si inicialmente `L`$\{2, 3, 4\}$ y `C`$\{1, 6, 3, 5, 2, 8\}$ entonces, después de hacer `SACA_PAR(L,C)` debe quedar `L`$\{2, 3, 4, 6, 2, 8\}$ y `C=`$\{1, 3, 5\}$. No usar ninguna estructura auxiliar más que una cola. Usar las funciones del `TAD LISTA: INSERTA(x,p,L)`, `RECUPERA(p,L)`, `SUPRIME(p,L)`, `SIGUIENTE(p,L)`, `ANULA(L)`, `PRIMERO(L)`, `FIN(L)` y del `TAD COLA: ANULA(C)`, `PONE_EN_COLA(x,C)`, `QUITA_DE_COLA(C)`, `VACIA(C)`, y `FRENTE_DE_COLA(C)`.

c) [**suma-par (25 puntos)**] Escribir una función `function SUMA_PAR(n: nodo; A: arbol) : integer` que retorna la suma de las etiquetas pares de un árbol binario. Usar las funciones del `TAD ARBOL BINARIO`: `HIJO_IZQ(n,A)}`, `HIJO_DER(n,A)`, `ETIQUETA(n,A)`.

Ej. 3.- [**LIBRES**] **Ejercicios operativos (total 80 puntos)**

a) [**colorear-grafo (20 puntos)**] Colorear el siguiente grafo, utilizando una estrategia heurística para tratar de usar el menor número de colores posibles.



b) [**reconstruir-arbol (20 puntos)**] Dibujar el árbol ordenado orientado cuyos nodos, listados en orden previo y posterior son

- `ORD_PRE` $=\{P, Q, R, T, U, S, V, W, Z\}$.

- `ORD_POST` $=\{T, U, R, V, Z, W, S, Q, P\}$.

c) **[abb (20 puntos)]** Dados los enteros $\{12, 11, 14, 8, 16, 13, 9, 7, 15, 21, 4\}$ insertarlos, en ese orden, en un *"árbol binario de búsqueda"*. Mostrar las operaciones necesarias para eliminar los elementos 12, 9 y 13.

d) **[heap-sort (20 puntos)]** Dados los enteros $\{3, 7, 1, 2, 5, 4, 3, 6\}$ ordenarlos por el método de *"montículos"* (*"heap-sort"*). Mostrar el montículo (minimal) antes y después de **cada** inserción/supresión.

Ej. 4.- [LIBRES] preguntas (total 20ptos, 5ptos/preg): [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) Dadas las funciones

- $T_1(n) = 5n + \log n$,
- $T_2(n) = 4n^2 + \sqrt{n}$,
- $T_3(n) = 2^n + n!$ y
- $T_4(n) = \sqrt{(n)} + \log n$

decir cuál de los siguientes ordenamientos es el correcto

- ☐ $T_3 < T_4 < T_1 < T_2$
- ☐ $T_4 < T_1 < T_2 < T_3$
- ☐ $T_2 < T_1 < T_4 < T_3$
- ☐ $T_4 < T_3 < T_2 < T_1$

b) El tiempo de ejecución para el algoritmo de clasificación por montículos (*"quicksort"*) es $O(n \log(n)$ (n es el número de elementos a ordenar) ...

- ☐ ... siempre.
- ☐ ... cuando el vector ya está ordenado.
- ☐ ... nunca.
- ☐ ... en el caso promedio.

c) Una ventaja del método de clasificación por selección, en comparación con otros algoritmos *lentos*, es que realiza sólo n *intercambios*...

- ☐ ... a veces.
- ☐ ... cuando el vector está ordenado.
- ☐ ... siempre.
- ☐ ... cuando el vector está desordenado.

d) ¿Cuál es el tiempo de ejecución del procedimiento de clasificación por *incrementos decrecientes* (*shell-sort*) en el caso promedio?

- ☐ $O(n^{1,3})$
- ☐ $O(n^{1,5})$
- ☐ $O(\log n)$
- ☐ $O(n!)$