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. [15 de Diciembre de 2005]

El examen se compone de dos partes.

- **Clases y programación: TODOS.**
- **Operativos y preguntas: Sólo LIBRES.**

Los libres deben cumplir con al menos un 70 % del porcentaje de la segunda parte.

1. [**todos: clases (20 puntos)**]. Escribir la implementación en `C++` del `TAD ARBOL BINARIO` (clase `btree`). Las funciones a implementar son `insert(p,x)`, `erase(p)` y `find(x)`. Observaciones:
   - En caso de optar por escribir la interfase *"básica"*, debe escribir todas las declaraciones necesarias de la clase, tanto en la parte privada como pública.
   - En caso de optar por la interfase *"avanzada"*, **no es necesario** declarar e implementar las clases `iterator` y `cell`.

2. [**todos: Ejercicios de programación**].

   a) [**nilpot (50 puntos)**]. Dadas dos correspondencias M_1 y M_2 la *"composición"* de ambas es la correspondencia $M = M_2 \circ M_1$ tal que si $M_1[a] = b$ y $M_2[b] = c$, entonces $M[a] = c$. Por ejemplo, si `M1={(0,1),(1,2),(2,0),(3,4),(4,3)}`, y `M2={(0,1),(1,0),(2,3),(3,4),(4,2)}`, entonces $M = M_1 \circ M_2$ `={(0,0),(1,3),(2,1),(3,2),(4,4)}`. Notemos que para que sea posible componer las dos correspondencias es necesario que los valores del contradominio de M_1 estén incluidos en las claves de M_2. Si el conjunto de valores del contradominio de una correspondencia M está incluido en el conjunto de sus claves, entonces podemos componer a M consigo misma, es decir $M^2 = M \circ M$. Por ejemplo, $M_1^2 = M_1 \circ M_1$`= {(0,2),(1,0),(2,1),(3,3),(4,4)}`. De la misma manera puede definirse, $M^3, ..., M^n$, componiendo sucesivamente. Puede demostrarse que, para algún n debe ser $M^n = I$, donde I es la *"correspondencia identidad"*, es decir aquella tal que $I[x] = x$. Por ejemplo, si `M = {(0,1),(1,2),(2,0)}`, entonces para $n = 3$, $M^n = M^3 = I$.
   Consigna: Escribir una función `int nilpot(map<int,int> &M);` que dada una correspondencia `M` retorna el mínimo entero `n` tal que $M^n = I$.
   Sugerencia: Escribir dos funciones auxiliares:
   - `void compose(map<int,int> &M1,map<int,int> &M2,map<int,int> &M);` que dadas dos correspondencias `M1, M2`, calcula la composición $M = M_2 \circ M_1$, devolviéndola en el argumento `M`,
   - `bool is_identity(map<int,int> &M);` que dada una correspondencia `M`, retorna `true` si `M` es la identidad, y `false` en caso contrario.

   b) [**elimina-valor (30 puntos)**].
   Escribir una función `void elimina_valor(queue<int>&C, int);` 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.*
   Restricciones: El algoritmo debe tener un tiempo de ejecución $O(n)$, donde n es el número de elementos en la cola original.

3. [**libres: Ejercicios operativos. (total 80 puntos)**].

   a) [**particionar (20 puntos)**]. Considerando el árbol `(f (a c e (h b)) (d g))` decir cuál son los nodos `descendientes(b)`, `antecesores(b)`, `izquierda(b)` y `derecha(b)`.

b) [**heap-sort (20 puntos)**]. Dados los enteros $\{5, 0, 8, 2, 4, 5, 2, 4, 1, 3\}$. ordenarlos por el método de *"montículos"* (*"heap-sort"*). Mostrar el montículo (minimal) **antes** y **después** de **cada** inserción/supresión.

c) [**huffman (20 puntos)**]. Dados los caracteres siguientes con sus correspondientes probabilidades, construir el código binario (para **todos** los caracteres) y encodar la palabra `FAVALORO`: $P(A) = 0.24$, $P(F) = 0.12$, $P(G) = 0.10$, $P(L) = 0.10$, $P(M) = 0.10$, $P(O) = 0.14$, $P(R) = 0.08$, $P(S) = 0.06$, $P(V) = 0.06$. Indicar el número de nivel de cada caracter y calcular la longitud promedio del código obtenido.

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

- `ORD_PRE`=$\{B, F, E, H, I, D, J, A, C, G\}$,
- `ORD_POST`=$\{E, F, I, D, A, J, C, H, G, B\}$.

4. [**libres: preguntas (20 puntos)**]. 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) Dado el árbol binario `(z (a b q) r)`, ¿cuál de las siguientes opciones es verdadera ?

- ☐ Es completo y es lleno.
- ☐ Es completo pero no lleno.
- ☐ Es lleno pero no completo.
- ☐ Ni es completo ni es lleno.

b) ¿Cuál es el tiempo de ejecución para intersección de conjuntos por vectores de bits? (N es el número de elementos en el conjunto universal, n el número de elementos en el conjunto dado)

- ☐ $O(N \log N)$
- ☐ $O(n)$
- ☐ $O(N)$
- ☐ $O(n \log n)$

c) Sea el árbol `(5 7 (8 6 9))`. Después de hacer:

```
n = D.find(7);
n++;
n = n.lchild();
n = n.lchild();
n = D.insert(n,2);
```

¿Cuál de las opciones es verdadera?

- ☐ `(5 7 (8 2 6 9))`
- ☐ Da un error.
- ☐ `(5 7 (8 6 2 9))`
- ☐ `(5 7 (8 (6 2) 9))`

d) ¿Cómo es el tiempo de ejecución para intercalar dos listas clasificadas de n elementos?

- ☐ ... $O(1)$
- ☐ ... $O(\log n)$
- ☐ ... $O(n)$
- ☐ ... $O(n!)$