Apellido y Nombre: _____

Carrera: _____ DNI: _____

[Llenar con letra mayúscula de imprenta GRANDE]

Universidad Nacional del Litoral
Facultad de Ingeniera y Ciencias Hdricas
Departamento de Informtica
Algoritmos y Estructuras de Datos

Algoritmos y Estructuras de Datos.
2do Parcial. [2013-11-14]

ATENCIÓN(1): Para aprobar deben obtener un **puntaje mínimo** del 60 % en las preguntas de teoría (Ej 4) y 50 % en las restantes secciones.
ATENCIÓN(2): Recordar que tanto en las clases (Ej. 1) como en los ejercicios de programación (Ej 2.) **deben usar la interfaz STL**.

[Ej. 1] [clases (20pt)]

a) **[AB (5pt)]** Par el TAD Arbol Binario (AB): declarar las clases `btree`, `cell`, `iterator`, incluyendo las declaraciones de datos miembros. Implementar el método `btree<T>::iterator btree<T>::find(const T& x);`. Si utiliza alguna función auxiliar impleméntela.

b) **[ABB (5pt)]** Escribir la función `pair<int,int> minmax(btree<int> &T);` que devuelve el mínimo y el máximo de el ABB `T`. Si el árbol está vacío debe devolver las constantes `(INT_MAX,INT_MIN)`.
Restricción: La complejidad algorítmica debe ser $O(l)$ donde l es la profundidad del árbol.

c) **[sort (10pt)]** Escribir una función `void sort(vector<int> &v);` que ordena los elementos de `v` utilizando el operador `<` como función de comparación. Puede elegir cualquier algoritmo de ordenamiento rápido o lento visto en el curso. **Especifique cuál es el algoritmo que está usando.**
Restricción: El algoritmo debe ser *in-place*, es decir no debe usar contenedores auxiliares.

[Ej. 2] [Programación (40pt)] Recordar que en los ejercicios de programación **deben usar la interfaz STL**.
Atención: Notar que hay 2 ejercicios cuya suma es **50**, pero el total de la sección es **40**. (O sea, la nota final en esta sección es `min(40,n1+n2)`, donde `n1,n2` son las notas de cada ejercicio).

a) **[count-if (20pt]** Escribir una función `int countif(btree<int> &T,bool (*pred)(int x));` que retorna el número de nodos del árbol `T` que satisfacen el predicado `pred`. Por ejemplo, si `T=(7 2 (4 (3 8 4) (5 . 6)))`, entonces `countif(T,odd)` debe retornar 3 y `countif(T,even)` debe retornar 5. Escribir las funciones predicados `odd` y `even`.

b) **[powerset (30pt)]**. Escribir una función `void powerset(set<int> &S,list<set<int> > &PS);` que dado un conjunto `S` devuelve el **conjunto potencia** de `S`. Por ejemplo, si `S={2,5}` entonces debe quedar `PS={{},{2},{5},{2,5}}`.
Ayuda: Escribir el algoritmo en forma **recursiva**.

- Si el conjunto es vacío, el único subconjunto es el conjunto vacío.
- Si no tomar un elemento cualquiera `x` del conjunto y tomar `W=S-{x}`.
- Formar `PW` todos los subconjuntos de `W` llamando recursivamente a la función.
- Formar `PS` tomando todos los elementos de `PW` y todos los de `PW` agregandole `x`. O sea `PS.size()=2*PW.size()`.

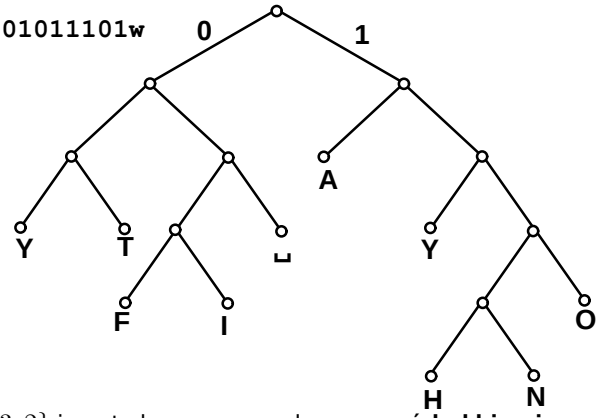
Por ejemplo, si `S={2,5}` entonces

- Sacamos cualquier elemento, por ejemplo `x=5`.
- Queda el conjunto `W={2}`.
- Los subconjuntos de `W` son `PW={{},{5}}`.
- Entonces `PS={{},{2},{5},{2,5}}`

[Ej. 3] [operativos (20pt)]

a) **[huffman (4pt)]** Dados los caracteres siguientes con sus correspondientes probabilidades, contruir el código binario utilizando el algoritmo de Hufmann y encodar la palabra `TIFON`
$P(F) = 0.10, P(N) = 0.10, P(O) = 0.10, P(T) = 0.30, P(i) = 0.05, P(C) = 0.35$. Calcular la longitud promedio del código obtenido.

b) **[hf-decode (4pt)]**
Utilizando el código de la derecha desencodar el mensaje
`00101010100111111101011111001001011101011101w`



c) **[abb (4pt)]** Dados los enteros $\{14, 8, 21, 3, 4, 11, 6, 5, 4, 13, 2\}$ insertarlos, en ese orden, en un **árbol binario de búsqueda (ABB)**. Mostrar las operaciones necesarias para eliminar los elementos 14, 8 y 3 en ese orden.

d) **[hash-dict (4pt)]** Insertar los números $\{3, 19, 29, 12, 11, 39, 21, 8\}$ en una **tabla de dispersión cerrada** con $B = 10$ cubetas, con función de dispersión `h(x) = x`.

e) **[heap-sort (4pt)]** Dados los enteros $\{4, 8, 11, 5, 6, 16, 13\}$ ordenarlos por el método de **montículos** (**heap-sort**). Mostrar el montículo (minimal) antes y después de cada inserción/supresión.

[Ej. 4] [Preguntas (total = 20pt, 4pt por pregunta)]

a) Explique porqué el método de Huffman cumple con la *"condición de prefijos"*.

b) Defina qué es un **ABC (Árbol Binario Completo)**. De ejemplos de AB que son ABC, y que no lo son.

c) Escriba el código para ordenar un vector de enteros `v` por **valor absoluto**, es decir escriba la función de comparación correspondiente y la llamada a `sort()`.
Nota: recordar que la llamada a `sort()` es de la forma `sort(p,q,comp)` donde `[p,q)` es el rango de iteradores a ordenar y `comp(x,y)` es la función de comparación.

d) Se quiere representar el conjunto de enteros múltiplos de 3 entre 30 y 99 (o sea $U = \{30, 33, 36, ..., 99\}$) por **vectores de bits**, escribir las funciones `indx()` y `element()` correspondientes.

e) Discuta el **número de inserciones** que requieren los algoritmos de ordenamiento lentos en el peor caso.