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.
Parcial 2. [2014-11-13]

ATENCIÓN: Para aprobar deben obtener un **puntaje mínimo** del 60 % en las preguntas de teoría (Ej 4) y 50 % en las restantes secciones.

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

Recordar que **deben usar la interfaz STL**.

a) **[aoo (5pt)]**
Arbol Ordenado Orientado: Implementar (haciendo uso de la interfaz STL) los métodos

- `iterator lchild();`
- `iterator right ();`
- `iterator insert (iterator p, T t);`

considerando que se debe escribir la clase `iterator` (en el scope que corresponda) además de aquellas estructuras sobre las cuales esta se construye (a saber, la clase `cell`).

b) **[hash-dict (5pt)]**
Diccionario por tablas de dispersión cerrada: Implementar la clase `iterator_t` en conjunto con aquello que considere pertinente dentro de la parte privada de la clase `hash_set`. Finalmente, implemente el método `iterator_t find (key_t & x)`

c) **[vbitset (5pt)]**
Conjuntos por vectores de bits: Implementar los métodos

- `pair_t insert (elem_t x);`
- `elem_t retrieve (iterator_t p);`
- `void set_difference (set & A, set & B, set & C);`

para ello deberá además implementar la parte privada de la clase.

d) **[abb (5pt)]**
Conjuntos por árbol binario de búsqueda: Implementar (haciendo uso de la interfaz STL) el método

- `void erase (iterator_m)`

definiendo para ello (y respetando el anidamiento pertinente) la clase iterator.

[Ej. 2] [operativos (W=20pt)]

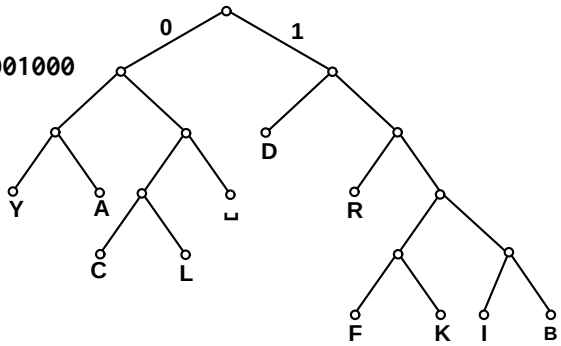
a) **[rec-arbol (3pt)]** Dibujar el AOO cuyos nodos, listados en orden previo y posterior son

- `ORD-PRE =(Z,A,B,C,D,E,F,T,G,H)`
- `ORD-POST =(A,E,F,D,G,H,T,C,B,Z)`

b) **[part-arbol (2pt)]** Dado el árbol ordenado orientado (AOO): `(A (R J L) (T (V (Z D) Q)))` determinar cuales son los nodos **antecesores, descendientes, izquierda** y **derecha** del nodo `V`. ¿Son **disjuntos**? Justifique.

c) **[huffman (3pt)]** Dados los caracteres siguientes con sus correspondientes probabilidades, contruir el código binario utilizando el algoritmo de Hufmann y encodar la palabra `CYBERMONDAY`,
$P(O) = 0.10$, $P(D) = 0.10$, $P(B) = 0.10$, $P(N) = 0.30$, $P(C) = 0.10$, $P(A) = 0.02$, $P(E) = 0.02$, $P(R) = 0.02$, $P(Y) = 0.04$, $P(M) = 0.20$.
Calcular la longitud promedio del código obtenido.

d) **[hf-decode (2pt)]**
Utilizando el código de la derecha desencodar el mensaje
`11111010100101001110101111100110111101000100 0`



e) **[abb (2pt)]** 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.

f) **[hash-dict (3pt)]** Insertar los números $\{2, 18, 28, 11, 10, 38, 20, 7\}$ en una **tabla de dispersión cerrada** con $B = 10$ cubetas, con función de dispersión **`h(x) = x`**.

g) **[heap-sort (3pt)]** Dados los enteros $\{3, 12, 10, 4, 5, 15, 7\}$ ordenarlos por el método de **montículos** (**heap-sort**). Mostrar el montículo (minimal) antes y después de cada inserción/supresión.

h) **[quick-sort (2pts)]** Dados los enteros $\{8, 2, 9, 4, 1, 3, 10, 2, 7, 5, 4, 0\}$ ordenarlos por el método de **clasficación rápida** (**quick-sort**). En cada iteración indicar el pivote y mostrar el resultado de la partición. Utilizar la estrategia de elección del pivote discutida en el curso, a saber el mayor de los dos primeros elementos distintos.

[Ej. 3] [Preguntas (W=20pt, 4pt por pregunta)]

a) Explique el concepto de **estabilidad** para algoritmos de ordenamiento. De ejemplos de algoritmos de ordenamiento estables y no estables.

b) Explique cuáles son las condiciones que debe satisfacer un predicado binario para ser una **relación de orden fuerte**.

c) Explique que las operaciones que realizan (semántica) las dos versiones de **`erase`** para conjuntos.

```
1 void erase(iterator p);
2 int erase(key_t x);
```

Explicar que son los valores de retorno, y porqué una retorna un entero y la otra no.

d) ¿Cuál es el tiempo de ejecución de **`find(x)`** para conjunto por árbol binario de búsqueda (mejor/promedio/peor)? Justifique.

e) ¿Porqué el árbol binario de una códificación binaria para compresión debe ser un árbol binario lleno (FBT)? (Recordemos que un FBT es áquel cuyos hijos son hojas o nodos interiores con sus dos hijos.)