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

Algoritmos y Estructuras de Datos.
2do parcial. [17 de Noviembre de 2016]

[Ej. 1] [Clases (mínimo 50 %)]

a) **[set-vb]:** Dada la siguiente interfaz para un conjunto (set) por vector de bits que contiene letras mayúsculas (de `A` a `Z`):

```
const int N=26;
typedef char elem_t;
int index(elem_t t);
elem_t element(int j);
typedef int iterator_t;
typedef pair<iterator_t,bool> pair_t;

class set {
private:
    vector<bool> v;
    iterator_t next_aux(iterator_t p);
public:
    set();
    pair_t insert(elem_t x);
    elem_t retrieve(iterator_t p);
    void erase(iterator_t p);
    bool erase(elem_t x);
    iterator_t find(elem_t x);
    iterator_t begin();
    iterator_t end();
    iterator_t next(iterator_t p);
};
```

implemente los métodos: `begin`, `next` e `insert`, y las funciones `index` y `element` (si dentro de alguno de ellos utiliza otro método de set, deberá implementarlo también).

b) **[set-abb]:** Dada la siguiente interfaz para un conjunto (set) por vector árbol binario de búsqueda:

```
template<class T>
class set {
private:
    typedef btree<T> tree_t;
    typedef typename tree_t::iterator node_t;
    tree_t bstree;
    ...
public:
    class iterator {
        note_t node;
        tree_t &bstree;
    private:
            friend class set;
            iterator(tree_t &t, node_t n) : bstree(t), node(n) {}
            ...
    };
    typedef pair<iterator,bool> pair_t;
```

```
    pair_t insert(T x);
    void erase(iterator m);
    int erase(T x);
    iterator find(T x);
    iterator begin();
    iterator end();
    ...
};
```

implemente los métodos `find`, `insert`, y `begin` (si dentro de alguno de ellos utiliza otro método de set, deberá implementarlo también).

[Ej. 2] [Operativos (mínimo 50 %)]

a) **[Huffman]** Dados los caracteres siguientes con sus correspondientes probabilidades, construir el código binario siguiendo el algoritmo de Huffman y encodar la palabra `ACLAMADA`.
P(A)=0.25, P(L)=0.1, P(R)=0.1, P(O)=0.15, P(M)=0.05, P(D)=0.05, P(S)=0.1, P(C)=0.15, P(E)=0.05
Calcular la longitud promedio del código obtenido. ¿Cual debería ser la probabilidad de cada caracter para minimizar la longitud del código de la palabra a encodar en el ejercicio?

b) **[quick-sort]:** Dados los enteros `{6, 10, 5, 2, 6, 11, 9, 4, 4, 3, 12, 7}` ordenarlos por el método de *clasificación 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.

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

d) **[hash]:** Insertar los enteros `{0, 19, 25, 8, 7, 35, 17, 4}` en una tabla de dispersión cerrada con `B=10` cubetas, con función de dispersión `h(x) = (x+1)%B`. Luego eliminar el elemento `7` e insertar el elemento `25`, en ese orden. Mostrar la tabla resultante.

[Ej. 3] [Preguntas (mínimo 60 %)]

a) Si queremos generar un código binario de longitud fija para el conjunto de las letras mayúsculas y minúsculas (52 caracteres). ¿Cuantos bits tendrá, como mínimo, cada caracter?

b) ¿Cuantos nodos tiene un árbol binario completo de 4 niveles?

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) Explique el concepto de estabilidad para algoritmos de ordenamiento.

e) ¿Cuál es el resultado de aplicar la función `l=particiona(w,j,k,v)`? (Nota: No se pide el algoritmo, sino cuál es el efecto de aplicar tal función, independientemende de como se programe). Explique los argumentos de la función. ¿Cuál debe ser el tiempo de ejecución para particiona() si queremos que quick_sort() sea un algoritmo rápido?

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

g) Discuta el tiempo de ejecución de los algoritmos de ordenamiento.

h) ¿Cuál es el tiempo de ejecución del algoritmo de ordenamiento rápido (quick-sort), en el caso promedio y en el peor caso? ¿Cuando se produce el peor caso?