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.
Super R-Recuperatorio. [2016-02-10]

[ATENCIÓN 1] Para aprobar deben obtener un **puntaje mínimo** del 60 % en las preguntas de teoría y 50 % en las restantes secciones.
[ATENCIÓN 2] Escribir cada ejercicio en **hoja(s) separada(s)**. Es decir todo `CLAS2` en una o más hojas **separadas**, `OPER2` en una o más hojas **separadas**, `PREG2` en una más hojas **separadas**, etc...
[ATENCIÓN 3] Encabezar las hojas con **sección, Nro de hoja (relativo a la sección), apellido, y nombre, ASI**: `CLAS2, Hoja #2/3 LOVELACE, ADA`

[Ej. 1] [CLAS1 (W=20pt)]

a) **[list]** Siendo la siguiente una posible implementación del TAD **Lista mediante celdas simplemente enlazadas por punteros**:

```
class cell;
typedef cell *iterator_t;
class list {
  private:
  cell *first;
  public:
  list();
  ~list();
  iterator_t insert(iterator_t p,elem_t j);
  iterator_t erase(iterator_t p);
  iterator_t erase(iterator_t p,iterator_t q);
  void clear();
  iterator_t begin();
  iterator_t end();
  iterator_t next(iterator_t p);
  iterator_t prev(iterator_t p);
  elem_t & retrieve(iterator_t p);
  int size();
};

class cell { ... };
```

Sin modificar la definición de la clase **`list`**,

- Complete la definición de la clase **`cell`**.
- Implemente el constructor y los métodos **`prev`** y **`size`**.
- Determine y justifique los órdenes de crecimiento $O(?)$ para los métodos implementados en el punto 2.

b) **[map]** Defina una clase **`map`** para modelar una correspondencia entre strings y doubles e implemente todos los métodos y auxiliares que considere necesarios para utilizarla en el siguiente ejemplo:

```
map nums;
nums.insert("golden ratio", 1.618 );
nums.insert("pi", 3.14159 );
nums.insert("ultimate answer", 42 );
nums.insert("speed of light", 299e8 );

string name;
cin >> name;
iterator_t it = nums.find(name);
if (it == nums.end()) cout << "Error: Number not found";
else cout << "Your number is " << nums.value(it);
```

Ayudas: se sugiere implementarla mediante un vector ordenado de structs (definir **`elem_t`**) , y definir (implementar) el método auxiliar **`lower_bound(...)`**. Nota: no hace falta declarar ni definir métodos u operadores de **`map`** que no se requieran para este ejemplo.

[Ej. 2] [CLAS2 (W=20pt)]

a) **[ab]** Para la siguiente implementación del **TAD Arbol Binario (AB)**:

```
class cell { /*...*/ };
class iterator_t {
/*...*/
public:
    iterator_t left() { /* ... */ }
    iterator_t right() { /* ... */ }
};
class btree {
private:
    cell *header;
    btree(const btree &T);
public:
    btree();
    ~btree();
    iterator_t insert(iterator_t p,elem_t t);
    iterator_t erase(iterator_t p);
    void clear();
    iterator_t begin();
    iterator_t end();
    bool empty();
    int size();
};
```

Complete las definiciones de las clases **`cell`** e **`iterator_t`** e implemente el constructor de la clase **`btree`** y el método **`insert`**.

b) **[hashset]** Para la siguiente implementación de un **diccionario por tabla de dispersión abiertas**:

```
typedef int key_t;
typedef int (*hash_fun)(key_t x);
class iterator_t {
    friend class hash_set;
private:
    int bucket;
    std::list<key_t>::iterator p;
    iterator_t(int b,std::list<key_t>::iterator q);
public:
    bool operator==(iterator_t q);
    bool operator!=(iterator_t q);
    iterator_t();
};
class hash_set {
private:
    typedef std::list<key_t> list_t;
    typedef list_t::iterator listit_t;
    typedef std::pair<iterator_t,bool> pair_t;
    hash_set(const hash_set&) {}
    hash_set& operator=(const hash_set&) {}
    hash_fun h;
    int B;
    int count;
    std::vector<list_t> v;
    iterator_t next_aux(iterator_t p);
public:
    hash_set(int B_a,hash_fun h_a)
        : B(B_a), v(B), h(h_a), count(0) {}
    iterator_t begin();
    iterator_t end();
    iterator_t next(iterator_t p);
    key_t retrieve(iterator_t p);
    pair_t insert(const key_t& x);
    iterator_t find(const key_t& x);
    int erase(const key_t& x);
    void erase(iterator_t p);
    void clear();
    int size();
};
```

Implemente los métodos **`insert(const key_t&)`** y **`erase(const key_t&)`**.

[Ej. 3] [PREG2 (W=20pt, 4pt por pregunta)]

a) ¿Es verdad que si dos nodos están en el **mismo nivel** de un árbol, entonces son **hermanos**? ¿Y la recíproca? De ejemplos.

b) ¿Es posible **insertar** en una posición **no-dereferenciable** (Λ) en un Árbol Binario (AB)? ¿Y en un Árbol Ordenado Orientado (AOO)? Discuta y de ejemplos.

c) Explique cual es la **condición de prefijos** para códigos binarios. De un ejemplo de códigos que cumplen con la condicion de prefijos y que no cumplen para un conjuntos de 3 caracteres.

d) Defina en forma recursiva el listado en **orden previo** y el listado en **orden posterior** de un **Árbol Ordenado Orientado (AOO)** con raíz **`t`** y sub-árboles hijos **`h_1,h_2,...,h_n`**.

e) Cual es el tiempo de ejecución (en notación $O(\cdot)$, en promedio) en función del número de elementos (n) que posee el árbol ordenado orientado implementado con celdas encadenadas por punteros de las siguientes operaciones/funciones/métodos

- **`insert(p,x)`**
- **`begin()`**
- **`lchild()`**
- **`operator++`** (prefijo/postfijo)

- `find(x)`
- `erase(p)`
- `*n`

f) Exprese como se calcula la longitud promedio de un código de Huffman en función de las probabilidades de cada unos de los caracteres P_i, de la longitud de cada caracter L_i para un número N_c de caracteres a codificar.

g) Para el arbol binario `(1 . (2 (3 5 .) 6))`

- como queda el árbol y que sucede si hacemos
```
btree<int>::iterator p=T.begin();
p=T.erase(p.left());
```
- y como queda el árbol asi hacemos por otro lado
```
btree<int>::iterator p=T.begin(),n;
n=p;
n=n.right();
p=T.splice(p.left(),n);
```

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

i) ¿Cuál es el costo de **inserción** en tablas de dispersión cerradas?

j) ¿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**?