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.
1er parcial. [04-10-2018]

[ATENCIÓN 1] Para aprobar deben obtener un **puntaje mínimo** del 60 % de la sección PREG. y 50 % en las restantes secciones.

[Ej. 1] [PREG (min 60 %)]

a) Explique qué quiere decir la propiedad de "Transitividad" de **`O()`**
b) Presente las diferencias entre las estrategias heurísticas y exhaustivas. Dé un ejemplo.
c) Comente ventajas y desventajas del uso de listas doblemente enlazadas con respecto a simplemente enlazadas. ¿Cuáles son los métodos cuyo tiempo de ejecución cambia y por qué?
d) ¿Por qué en un mapa en el que el método **`find`** es **`O(log(n))`**, el método **`retrieve(x)`** puede ser **`O(n)`**? ¿En qué casos ocurre?
e) Dentro de la clase **`tree`** que define un AOO, ¿por qué no es suficiente definir al iterador como **`typedef iterator *cell`**? Explique
f) Sea el AOO **`T=(A B C (D E))`**, y tomando dos iteradores **`it1 = T.find(B).lchild()`** e **`it2 = T.find(D).right()`**. Grafique el árbol y ubique los iteradores mencionados presentando claramente los punteros que contienen. ¿Por qué a pesar de las diferencias obtengo verdadero al hacer **`it1==it2`**?.

[Ej. 2] [CLASES (min 50 %)]

a) **[s-list]** Dada la siguiente implementación para una lista simplemente enlazada:

```
class cell {
    int elem;
    cell *next;
    friend class list;
};
using iterator = cell*;
class list {
    cell *first, *last;
public:
    list();
    ~list();
    iterator insert(iterator p, int x);
    iterator erase(iterator p);
    void clear();
    iterator begin();
    iterator end();
    iterator prev(iterator p);
    iterator next(iterator p);
    int retrieve(iterator p);
    int size();
    void splice(iterator p, list &l2, iterator from, iterator to);
    friend void swap(list &l1, list &l2);
};
```

implementar las funciones: **`list::retrieve(...)`**, **`list::size()`**, **`list::splice(...)`**, y **`swap(...)`** de forma tal que **`list::splice`** y **`swap`** sean **`O(1)`**. Nota: recuerde que splice mueve una parte de **`l2`**, definida por **`[from,to)`**, hacia la posición **`p`** de la lista que hace la llamada.

b) **[map-vo]** Considerando la siguiente definición de una clase **`map`** que mapea enteros a strings utilizando un vector ordenado:

```
class map {
    vector<pair<int,string>> v;
    iterator lower_bound(int x);
    ...
public:
```

```
    ...
    iterator find(int x);
    pair<bool,iterator> insert(int x);
    string retrieve(int x);
    ...
};
```

defina el tipo iterator, e implemente el método **lower_bound** de manera tal que sea **O(log(n))**; y utilícelo luego para implementar también **find**, **insert** y **retrieve**.

[Ej. 3] [OPER (min 50 %)]

a) **[coloreo]** Un paseador de perros tiene 10 clientes: Huesos, Coraje, Procer, Oddie, Scooby, Pluto, Brian, Snoopy, Bolt y Goofy. Pero no puede pasear a los 10 perros a la vez porque hay perros que se sabe que se pelean entre sí. Huesos siempre quiere morder a Procer y a Oddie. Scooby odia a Pluto y a Goofy. Coraje les tiene mucho miedo a Goofy, Oddie y Snoopy. Brian no quiere pasear ni con Bolt ni con Oddie. Y finalmente, se sabe que Bolt tampoco se lleva bien con Procer. Cómo utilizaría coloreado de grafos para modelar este problema y armar grupos de paseo. Utilizando la estrategia que proponga, genere una coloración válida y explique qué significa el resultado.

b) **[draw]** Dibujar el AOO cuyos nodos, listados en orden previo y posterior son

- **ORD-PRE = (C B F C I H G A J D E)**,
- **ORD-POST = (F I C B H A D J E G C)**,

luego particionar el conjunto de nodos respecto al nodo **J**.

c) **[orden]** Ordenar las siguientes funciones por tiempo de ejecución:

$$T_1(n) = 5\sqrt{n^3} + \log(5n)$$
$$T_2(n) = 3n^2 + 10^4$$
$$T_3(n) = 3^n + 3n^3$$
$$T_4(n) = 5n^2 \log(n)$$

Además, para cada una de las funciones **T1 , . . . , T4** determinar su velocidad de crecimiento.

d) **[function-f]** Dada la siguiente función y su *wrapper*:

```
void f(int n, vector<bool>& VL){
    if(n==0){
        for(auto x:VL) cout<<x<<" ";
        cout<<endl;
    }else{
        auto VL_0 = VL;
        auto VL_1 = VL;
        VL_0.push_back(0);
        VL_1.push_back(1);
        f(n-1,VL_0);
        f(n-1,VL_1);
    }
}
void f(int n){
    vector<bool> VL;
    f(n,VL);
}
```

- Renombrar la función con un nombre alusivo a la tarea que efectúa.
- Determinar el orden algorítmico de **f** respecto al parámetro **n**.