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.
Recuperatorio. [2018-11-20]

1. **[ATENCIÓN 1]** Para aprobar deben obtener un **puntaje mínimo** del 60 % en las preguntas de teoría y 50 % en clases y operativos.

2. **[ATENCIÓN 2]**

   - Escribir cada ejercicio (sección) 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...
   - Encabezar las hojas con **sección, Nro de hoja (relativo a la sección), apellido, y nombre, ASI**:
     `CLAS1, Hoja #2/3` `TORVALDS, LINUS`
   - En caso de no entregar una sección, agregar una hoja vacía con el nombre.
   - Al entregar, verificar con el docente que recibe que todas las secciones estén entregadas en forma correcta.

[Ej. 1] [CLAS1 (W=20pt)]
Recordar que **deben usar la interfaz STL**.

a) **[s-list]** Dada la siguiente definición 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 begin();
    iterator end();
    iterator prev(iterator p);
    iterator next(iterator p);
    int retrieve(iterator p);            //<implementar
    iterator insert(iterator p, int x);  //<implementar
    iterator erase(iterator p);
    void clear();
    int size();
    void splice(iterator p, list &l2, iterator from, iterator to); //<implementar
};
```

implementar los método: `retrieve(...)`, `insert(...)` y `splice(...)` (de forma tal que `splice` sea `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) **[AOO]:** Dada la siguiente implementación de árbol ordenado orientado:

```
class cell { ... };         //< definir

class iterator_t { ... };   //< definir

class tree {
  private:
   cell *header;
   tree(const tree &T);
  public:
   tree();
   ~tree();
   elem_t &retrive(iterator_t p);
   iterator_t insert(iterator_t p,elem_t elem);  //< implementar
```

```
14    iterator_t erase(iterator_t p);              //< implementar
15    void clear();
16    iterator_t begin();
17    iterator_t end();
18    ...
19 }
```

Definir las clases **cell** e **iterator** e implementar los métodos **insert(...)**, **erase** y **find(...)** de la clase **tree**.

[Ej. 2] [OPER1 (W=20pt)]

a) **[AOO (5pt)]** 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**.

b) **[orden (5pt)]** Dada la siguiente función y su *wrapper*:

```
1 void f(int n, vector<bool>& VL){
2     if(n==0){
3         for(auto x:VL) cout<<x<<" ";
4         cout<<endl;
5     }else{
6         auto VL_0 = VL;
7         auto VL_1 = VL;
8         VL_0.push_back(0);
9         VL_1.push_back(1);
10        f(n-1,VL_0);
11        f(n-1,VL_1);
12    }
13 }
14 void f(int n){
15    vector<bool> VL;
16    f(n,VL);
17 }
```

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

c) **[coloreo (5pt)]** Estamos organizando un congreso de informáticos y cualquier hotel de la ciudad tiene capacidad para alojar a todos los informáticos, pero hay algunos que están peleados a tal punto que no pueden alojarse en el mismo hotel. Mencione como modelaría el problema de asignar hotel a cada participante por medio de un problema de coloreo de grafos. Indique que representan los vértices, las aristas y los colores.

d) **[size (5pt)]** Suponga que cierto procesador tiene una frecuencia de reloj de 1 Ghz, y simplifique el análisis suponiendo que por cada diez ciclos de reloj se realiza una operación en particular. Estime de qué tamaño puede ser el problema que se puede resolver en diez segundos usando un algoritmo que requiere $T(n)$ operaciones, con los siguientes valores de $T(n)$: $\log(n)$, n y 2^n.

[Ej. 3] [PREG1 (W=20pt)]

a) Ordenar las siguientes funciones por tiempo de ejecución. Además, para cada una de las funciones $T_1, \ldots, T_5$ determinar su velocidad de crecimiento (expresarlo con la notación $O(\,\cdot\,)$).

$$T_1 = 3\,n^4 + 2\log_{10} n + \sqrt[4]{n}$$
$$T_2 = 2 \cdot 3^n + 3\,n! + 4\,n^2$$
$$T_3 = 3^2 + 3n^2 + 3 \cdot 3^n$$
$$T_4 = \log_{10} n + 4$$
$$T_5 = \log(13) n^{3.5} + 3.4 \log n + 2^4$$

b) ¿Como se define la altura de un nodo en un árbol? ¿Cuál es la altura de los nodos **o**, **p**, y **q** en el AOO siguiente: **(k (o p (q (r t u v))) l m)**?

c) Explique la propiedad de transitividad de la notación asintótica $O(\cdot)$. De un ejemplo.

d) ¿Cuáles son los tiempos de ejecución para los diferentes métodos de la clase **map<>** implementada con **listas ordenadas** y **vectores ordenados** en el caso promedio? Métodos: **find(key)**, **M[key]**, **erase(key)**, **erase(p)**, **begin()**, **end()**, **clear()**.

e) Sea el árbol **(a b q (z r (p t)))**. Cuáles de los siguientes son **caminos**?

- **(a b)**
- **(b a)**
- **(z p t)**
- **(r z p)**

f) ¿Existe una relación biunívoca entre un Arbol Ordenado Orientado (AOO) y su **orden previo**? ¿Y con el **orden posterior**? ¿Y con ambos a la vez? De ejemplos.

g) Explique qué se entiende por algoritmos de **búsqueda exhaustiva** y **heurístico**. Discuta cuales son las ventajas y desventajas.

h) Exprese la **regla del producto** para la notación $O(.)$. De un ejemplo.

[Ej. 4] [CLAS2 (W=20pt)]

a) **[set-tda]** Dada la siguiente base para una clase que representa un conjunto mediante una tabla de disperción abierta:

```
class iterator_t { ... };                  //< definir
typedef int (*hash_func_t)(key_t x);
class hash_set {
   vector<list<key_t>> v;
   hash_func_t h;
   int B;
public:
   hash_set(int B, hash_func_t f);
   iterator_t  begin();                    //< implementar
   iterator_t  end();                      //< implementar
   iterator_t  next(iterator_t p);         //< implementar
   key_t retrieve(iterator_t p);
   pair<iterator_t,bool> insert(key_t x); //< implementar
   iterator_t find(key_t x);
   int erase(key_t x);
   void erase(iterator_t p);
};
```

Definir la clase **iterator_t**. e implementar los métodos **begin**, **end**, **next** e **insert** de la clase **hash_set** (si dentro de estos métodos utiliza algún otro método o función auxiliar, también deberá implementarlo/a).

b) **[AB]:** Dada la siguiente implementación de árbol binario:

```
class cell {
    friend class btree;
    friend class iterator;
    elem_t elem;
    cell *right, *left;
};

class iterator {
    friend class btree;
    cell *ptr, *father;
    enum side_t (NONE,R,L);
    side_t side;
    iterator(cell *p, side_t side_a, cell *f_a);
  public:
    ...
};
```

```
class btree {
  private:
    cell *header;
    ...
  public:
    btree();
    ~btree();
    elem_t & retrieve(iterator p);
    iterator find(iterator p, elem_t e);   //<
    iterator insert(iterator p, elem_t e); //<
    iterator erase(iterator p);            //<
    void clear();
    iterator begin();
    iterator end();
    ...
};
```

- Implementar los métodos `insert(...)`, `erase(...)` y `find(...)`.

[Ej. 5] [OPER2 (W=20pt)]

a) **[Huffman (5pt)]** Se procesa un texto y se extrae la siguiente probabilidad de ocurrencia de cada caracter: `P({char,prob}) = [{'A', 0.15}, {'C', 0.1}, {'E', 0.1}, {'H', 0.05}, {'L', 0.1}, {'M', 0.05}, {'N', 0.1}, {'O', 0.15}, {' ', 0.2}]`. Hallar una codificación Huffman e indicar su longitud promedio. Luego encodar: `'EL CLAN'`.

b) **[quick-sort (5pt)]** Ordenar los enteros `{3, 1, -5 -2, 6, 9, 4, -1}` utilizando el algoritmo quick-sort. Indicar claramente las iteraciones del algoritmo (seleccion de pivot y partición en listas).

c) **[hash (5pt)]** Insertar los enteros `{-2, 7, 5, -12, 7, 33, 12}` en una tabla de dispersión cerrada con B=9 cubetas, con función de dispersión `h(x) = x%B` y redispersión linear. Luego eliminar el elemento `-2` e insertar el elemento `25`, en ese orden. Mostrar la tabla resultante.

d) **[abb (5pt)]** Dados los enteros `{4,-8,-23,6,42,-15,37,16}` insertarlos, en ese orden, en un árbol binario de búsqueda. Mostrar las operaciones necesarias para eliminar los elementos 4, -15 y -23 en ese orden.

[Ej. 6] [PREG2 (W=20pt)]

a) Explique cual es la condición de códigos prefijos. De un ejemplo de códigos que cumplen con la condicion de prefijo y que no cumplen para un conjuntos de 3 caracteres.
b) 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.
c) Comente ventajas y desventajas de las **tablas de dispersión abiertas y cerradas**.
d) Explique 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.
e) Explique el concepto de estabilidad para algoritmos de ordenamiento.
f) ¿Cómo se define en forma recursiva el **orden posterior** para un **árbol ordenado orientado (AOO)**?
g) Escriba las instrucciones necesarias para crear el siguiente **árbol binario (AB)**: `T=(1 (2 . 3) .)`
h) ¿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.)