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

FICH

UNL

Algoritmos y Estructuras de Datos.
Recuperatorio. [2016-11-24]

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 en **hoja(s) separada(s)**. Es decir todo `CLAS1` en una o más hojas **separadas**, `OPER1` en una o más hojas **separadas**, `PREG1` en una más hojas **separadas**, etc...
3. **[ATENCIÓN 3]** Encabezar las hojas con **sección, Nro de hoja (relativo a la sección), apellido, y nombre, ASI**: CLAS1, Hoja #2/3 TORVALDS, LINUS

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

a) **[map-vo]** Considerando la siguiente definición parcial de una clase map que mapea enteros a strings:

```
class map {
  vector<pair<int,string>> v;
  ...
  public:
  ...
  Iterator find(int x);
  pair<bool,iterator> insert(int x);
  int erase(int x);
  ...
};
```

defina el tipo `iterator`, e implemente los métodos `find`, `insert` y `erase` de manera tal que todas estas operaciones sean $O(\log(n))$. Si dentro de alguno de ellos utiliza otro método de `map`, deberá implementarlo también.

b) **[aoo]** Dada la siguiente definición de una clase para representar un árbol ordenado orientado:

```
class tree {
  cell *header;
  ...
  public:
  tree();
  ~tree();
  elem_t &retrieve(iterator_t p);
  iterator_t insert(iterator_t p,elem_t elem);
  iterator_t erase(iterator_t p);
  iterator_t splice(iterator_t to,iterator_t from);
  iterator_t find(elem_t elem);
  void clear();
  iterator_t begin();
  iterator_t end();
  ...
};
```

defina las clases `cell` e `iterator_t` e implemente todos sus métodos (`lchild`, `right` y operadores `!=` y ==).

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

a) **[particiona]** Dado el AOO cuyo listado previo es `(L M R W P J S H B K N)` y posterior es `(R P W M H B K S N J L)`, determine los antecesores propios, descendientes propios, izquierda y derecha del nodo `S`. ¿Son conjuntos disjuntos? Justifique.

b) **[coloreo]** El departamento informático de la universidad se compone de los 6 comités que se Enumeran a continuación:

```
C1 = {Pablo, Mario}
C2 = {Romina, Jose, Pablo}
C3 = {Marcos, Mario}
C4 = {Laura, Juan}
C5 = {Jose, Mario}
C6 = {Pedro, Juan, Marcos}
```

Estos comités se reúnen un día específico al mes. ¿Existe una mejor solución que plantear 6 horarios diferentes para cada reunión de tal manera de que nadie tenga programada dos reuniones en simultáneo? Encuentre una mejor propuesta por medio de una coloración del grafo que modela el problema.

c) **[aoo]** Utilizando sólo los métodos **`insert()`**, **`right()`** y **`lchild()`** del TAD AOO y sus iteradores, complete el siguiente código para armar el árbol **`T=(4 9 (7 -1 2) 11)`**.

```
tree<int> T;
tree<int>::iterator node = T.insert(T.begin(), 4);
. . .
```

d) **[orden]** La siguiente función evalúa un polinomio cuyos coeficientes son **a** es decir $p(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$ y posee un método **`evalua`** el cual retorna el valor del polinomio evaluado con **`x=c`**, esto es **`p(c)`**;

```
double evalua_1 (vector<int> &a,double c) {
  double resultado= 0.0;
  for (int termino= 0; termino < a.size(); termino++) {
    double xn= 1.0;
    for (int j=0; j < termino; j++)
      xn *= c;          // x elevado a n
    resultado += a[termino] * xn;
  }
  return resultado;
}
```

Responda:

1) De qué orden algorítmico es el método **`evalua_1`**?
2) Proponga un método **`evalua_2`** que sea más eficiente reduciendo la complejidad algorítmica.

[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 $O(\,\cdot\,)$).

$$T_1 = 2\log_{10} n + \sqrt[3]{n} + 4\,n^4 + 6\,n^2$$
$$T_2 = 2\,n^4 + 2\,n! + 3 \cdot 4^n$$
$$T_3 = 4 \cdot 4^n + 3^3 + 4n^3$$
$$T_4 = \log_2 n + 3$$
$$T_5 = 3.3\log n + 4^4 + \log(25)n^{2.5}$$

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

c) ¿Cuáles son los tiempos de ejecución para los diferentes métodos de la clase **`list<>`** implementada con **celdas simplemente enlazadas** (por punteros o cursores) en el caso promedio? *Métodos:* **`insert(p,x)`**, **`*p`**, **`erase(p)`**, **`clear()`**, **`begin()`**, **`end()`**.

d) ¿Es la **pila** un contenedor **FIFO** o **LIFO**? ¿Que significa? Responda la misma pregunta para el TAD **cola**.

e) ¿Cual es la principal ventaja de implementar correspondencias con **vectores ordenados**? ¿Se obtiene la misma ventaja con **listas ordenadas**?

f) Discuta si los algoritmos de **búsqueda exhaustiva** y **heurístico** para la **coloración de grafos** dan la solución **óptima** al problema. Señale la complejidad algorítmica de los mismos. ¿Cuál de los dos elegiría para un grafo con 100 vértices y 500 aristas? Justifique.

g) ¿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()`**.

h) Defina que es un **camino** en un árbol. Dado el AOO **`(z x (y (d e f)))`**, cuáles de los siguientes son caminos?

1) **`(y,d,e)`**,
2) **`(d,y,z)`**,
3) **`(e,d,f)`**,
4) **`(d,f)`**.

i) ¿Es posible **insertar** en una posición **no-dereferenciable** (Λ) en un árbol ordenado orientado (AOO)? ¿Y en una posición **dereferenciable**?

j) ¿Como se define la **altura** de un nodo en un árbol? ¿Cuál es la altura de los nodos **`A`**, **`B`**, y **`C`** en el siguiente AOO: **`(C (B (T X Y)) (Z A (D P)))`**?

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

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

a) **[ab]** Siendo la siguiente una posible implementación de la clase btree, que representa un árbol binario:

```
class cell {
  friend class btree;
  friend class iterator_t;
  elem_t t;
  cell *right,*left;
};
class iterator_t {
  friend class btree;
  cell *ptr,*father;
  enum side_t {NONE,R,L};
  side_t side;
  iterator_t(cell *p,side_t side_a,cell *f_a);
public:
  iterator_t();
  bool operator!=(iterator_t q);
  bool operator==(iterator_t q);
  iterator_t left();
  iterator_t right();
};
class btree {
private:
  cell *header;
  ...
  public:
  btree();
  ~btree();
  elem_t & retrieve(iterator_t p);
  iterator_t insert(iterator_t p,elem_t t);
  iterator_t erase(iterator_t p);
  void clear();
  iterator_t begin();
  iterator_t end();
  ...
}
```

implemente el constructor de la clase **`btree`** y los métodos **`insert`** y **`erase`**.

b) **[set-lo]** Dada la siguiente interfaz de una clase set que implementa un conjunto mediante una lista ordenada:

```
class set {
  list<elem_t> L;
public:
  elem_t retrieve(iterator_t p);
  iterator_t lower_bound(elem_t t);
  iterator_t next(iterator_t p);
  pair<iterator_t,bool> insert(elem_t x);
  void erase(iterator_t p);
  void erase(elem_t x);
  void clear();
  iterator_t find(elem_t x);
  iterator_t begin();
```

```
    iterator_t end();
    int size();
    friend void set_union(set &A,set &B,set &C);
    friend void set_intersection(set &A,set &B,set &C);
    friend void set_difference(set &A,set &B,set &C);
};
```

Implementar las funciones **set_union**, **set_intersection** y **set_difference** de forma que sean todas $O(n)$.

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

a) **[huffman]** Al procesar un archivo de texto de 500 caracteres se contabilizaron 100 "**A**", 50 "**I**", 50 "␣" (espacios o caracteres en blanco), 75 "**R**", 25 "**P**", 100 "**O**", 50 "**E**", 25 "**S**" y 25 "**L**". Se pide que se construya el árbol de la codificación de Huffman óptima para ese texto y se codifique la frase **EL␣PERRO**. ¿Cuál es la longitud promedio de la codificación obtenida?

b) **[hash]** Insertar los enteros **{3, 7, 13, 25, 9, 11, 7, 17}** en una tabla de dispersión cerrada con **B=12** cubetas, con función de dispersión **h(x) = x%B** y función de redispersión **d(x) = x-1**. Luego elimine el elemento 25 e inserte el elemento 4. Muestre la tabla resultante en cada paso.

c) **[abb]** Dados los enteros **{15, 7, 11, 23,18, 19, 9, 30, 1}** insertarlos en ese orden en un árbol binario de búsqueda. Mostrar las operaciones necesarias para eliminar los elementos 23 y 7 en ese orden.

d) **[quick-sort]** Dados los enteros **{17, 6, 5, 9, 11, 16, 22, 1, 11, 18}** 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 entre los dos primeros elementos distintos.

[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) Se quiere representar el conjunto de enteros múltiplos de 3 entre 30 y 99 (o sea $U = \{30, 33, 36, ..., 99\}$) por **vectores de bits**, escribir las funciones **indx()** y **element()** correspondientes.

e) Discuta la complejidad algorítmica de las operaciones binarias **set_union(A,B,C)**, **set_intersection(A,B,C)**, y **set_difference(A,B,C)** para conjuntos implementados por vectores de bits, donde **A**, **B**, y **C** son subconjuntos de tamaño n_A, n_B, y n_C respectivamente, de un conjunto universal **U** de tamaño N.

f) ¿Cuál es el costo de **inserción exitosa** en tablas de dispersión abiertas?

g) Se quiere representar conjuntos de pares de enteros en el rango **[0,10)** es decir por ejemplo **(3,7)**, **(9,0)**, por **vectores de bits**. ¿Cuál es el tamaño **N** del conjunto universal? Completar las funciones **indx()** y **element()** correspondientes:

```
1       int indx(pair<int,int> p) { /* ... */ }
2       pair<int,int> element(int z) { /* ... */ }
```

h) Si la correspondencia **map<int,string> M** contiene **M={1->"Romero",6->"Biglia"}** y ejecutamos el código **string z= M[6]**. ¿Que valor toma **z**? ¿Cómo queda **M**? Si ahora hacemos **z = M[10]**, ¿cómo quedan **z** y **M**?

i) Si queremos generar un código binario de longitud fija para el conjunto de letras minúsculas, mayúsculas, y dígitos (en total 26+26+10=62 caracteres. ¿Cuantos bits tendrá, como **mínimo**, la representación de cada caracter? ¿Y para el conjunto de caracteres mayúsculas y dígitos (26+10=36)?

j) ¿Es posible **insertar** en una posición **no-dereferenciable** (Λ) en un árbol binario (AB)? ¿Y en una posición **dereferenciable**?