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.
Parcial 1. [2019-10-03]

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 `CLAS2` en una o más hojas **separadas**, `OPER2` en una o más hojas **separadas**, `PREG2` 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)]

a) Defina la estructura de la clase celda de un AOO (`tree<T>::cell`). Presente los atributos y el prototipo de los métodos principales.

b) Defina la estructura de la clase iterador de un AOO (`tree<T>::iterator`). Presente los atributos y el prototipo de los métodos principales.

c) Dado el siguiente fragmento de código:

```
1 tree<char> T;
2 lisp2tree("(a (h (k s)))", T);
3 tree<char>::iterator p = T.begin().lchild().right();
```

En base a *a)* y *b)*, presente un croquis de todas las estructuras `tree<T>::cell` e `tree<T>::iterator` presentes en memoria luego de la ejecución y los punteros que las relacionan.

d) Defina la estructura de la clase celda de una lista simplemente enlazada (`list<T>::cell`). Presente los atributos y el prototipo de los métodos principales.

e) Implemente el constructor por defecto de la clase `list` (simplemente enlazada)

f) Implemente el método

```
1   typedef list<T>::iterator it_t;
2   it_t list<T>::splice(it_t it, list<T>& L,it_t first,it_t last);
```

el cual inserta el rango de elementos `[first, last)` de la lista `L` en la lista `this` utilizando un procedimiento $\mathcal{O}(1)$. Considere listas simplemente enlazadas.

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

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

```
1       ORD-PRE = (C M J K R L A N P F) ,
2       ORD-POST = (J M L N A R F P K C) ,
```

luego particionar el conjunto de nodos respecto al nodo `R`.

b) Un profesor de historia de una carrera a distancia quiere evaluar a sus alumnos presentándole a cada uno de ellos un texto para analizar para que luego respondan un cuestionario de tipo *multiple choice*. En un intento por evitar que dos alumnos se copien, el profesor ha investigado a través de las redes sociales qué alumnos se conocen entre sí; y ahora le pide ayuda a usted para no tener que hacer un enunciado diferente por cada alumno. Modele este problema con un grafo de forma tal que pueda aplicar coloreado de grafos para determinar un número razonable de enunciados diferentes a preparar y cómo distribuirlos entre los alumnos.

c) **[operaciones]**: Sea el árbol `D=(5 (4 3 6) (1 9 8 7) (4 6))`. Después de hacer:

```
auto n = D.find(1);
n++;
n = n.lchild();
n = n.lchild();
n = D.insert(n,2);
```

¿Cómo queda el árbol? ¿Se produce un error?

d) **[Notación $O(\cdot)$]**. cada una de las funciones $T_1, \ldots, T_4$ determinar su velocidad de crecimiento (expresarlo con la notación $O(\cdot)$) y ordenarlas de forma creciente.

$$T_1 = n^2\,(2n + 4\,n^2) + 2\log_{10} n + \sqrt[2]{n}$$
$$T_2 = 4\,n! + 2\,n^4 + 4 \cdot 3^n$$
$$T_3 = 4 \cdot 4^n + 4^4 + 2n^4$$
$$T_4 = 4^4 + 4.2\log n + \log(15)n^{3.5}$$

e) **[function-f]**: Renombrar la siguiente función con un nombre alusivo a la tarea que efectúa y determinar su orden algorítmico respecto al tamaño del vector **v**.

```
int f(vector<int>& v, int x){
  int i = 0;
  int j = v.size()-1;
  while(i<=j){
    int r = (i+j)/2;
    if(x==v[r]) return r;
    if(x>v[r]){
      i = r+1;
    }else{
      j = r-1;
    }
  }
  return -1;
}
```

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

a) Sea el árbol **(a (b t) (e (q r z) f))**.
- Cuáles de los siguientes son **caminos**?
  - **(b t)**
  - **(r q z)**
  - **(e q z)**
  - **(f e a)**
- Liste las **hojas** del árbol y los nodos a **profundidad** 2.
- ¿Cuál es la **altura** del árbol?

b) Sea la **correspondencia** **M={(6->5),(12->9)}** y ejecutamos el código int **x= M[12]**. ¿Qué ocurre? ¿Qué valores toman **x** y **M**? ¿Y si hacemos **x = M[5]**?

c) Discuta las ventajas y desventajas de utilizar listas **doblemente enlazadas** con respecto a las **simplemente enlazadas**. ¿Cuáles son los métodos cuyo tiempo de ejecución cambia y por qué?

d) ¿Cuál es la **complejidad algorítmica** (mejor/promedio/peor) de las siguientes funciones? (asumir listas implementadas con celdas enlazadas por punteros o cursores)

1) **list<T>::end(),**
2) **list<T>::erase(p),**
3) **list<T>::clear(),**
4) **map<K,V>::find(x),** para la implementación con vectores ordenados,
5) **map<K,V>::find(x),** para la implementación con listas ordenadas.

e) ¿Qué ventajas o desventajas tendría implementar la clase **stack** (pila) en términos de **lista simplemente enlazada** poniendo el tope de la pila en el fin de la lista? ¿Cuáles serían los tiempos de ejecución para **`top`**, **pop**, **`push`**? Lo mismo para cola: ¿Qué ventajas o desventajas tendría implementar la clase **queue** (cola) en términos de **lista simplemente enlazada** poniendo el frente de la cola en el fin de la lista? ¿Cuáles serían los tiempos de ejecución para **`front`**, **pop**, **`push`**?