Algoritmos y Estructuras de Datos – Curso 2011 (Redictado)
Parcial – 1era Fecha - Viernes 16 de Diciembre

Ejercicio 1.

Dada la siguiente recurrencia,

1.- Calcular el T(n) resolviendo la recurrencia, detallando los pasos seguidos para llegar al resultado.

2.- Calcular el O(n) justificando usando la definición de big-OH.

$$T(n) \begin{cases} 4 & n = 1 \\ 2T(n/2) + 5n + 1 & n > 1 \end{cases}$$

En el caso de ser necesario tenga presente las siguientes series:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$$

$$\sum_{i=1}^{n} i^4 = \frac{n(n+1)\ (6n^3+9n^2+n-1)}{30} \qquad \sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1}.$$

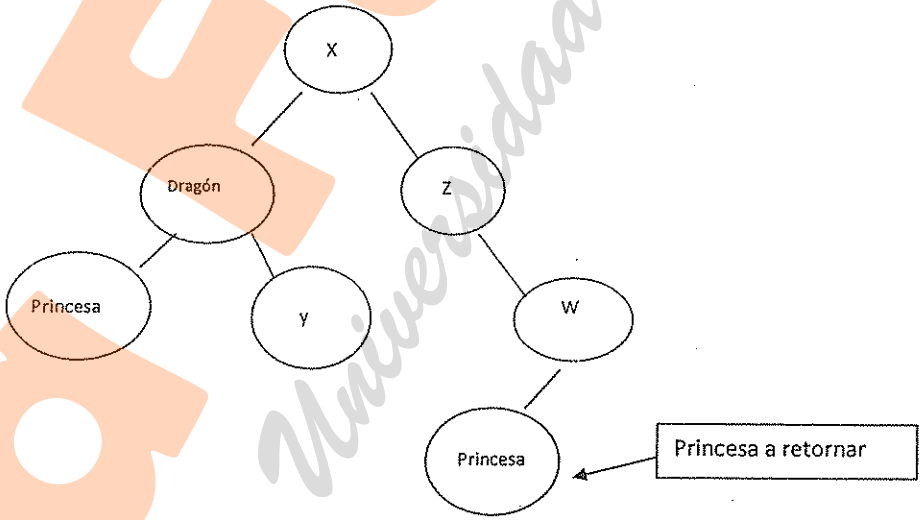
Ejercicio 2.

Supongamos disponible un tipo **Personaje** con las siguientes operaciones:

Personaje
- String tipo - String nombre
+esDragon (): boolean +esPrincesa (): boolean +setTipo (String unTipo)

Las operaciones **esDragon (): boolean** y **esPrincesa (): boolean** permiten averiguar si un personaje dado es un dragón o una princesa, respectivamente.
Suponemos que ningún personaje es dragón y princesa a la vez, y que un personaje puede no ser ninguna de las dos cosas.
Dado un árbol binario de personajes, se denominan nodos **accesibles** a aquellos nodos tales que a lo largo de la raíz hasta el nodo (ambos inclusive), NO se encuentra ningún dragón.
Debe implementar un método princesaAccesible():Personaje en la clase árbol binario que encuentre una princesa **accesible** lo más cerca posible de la raíz de un árbol dado,

Supongamos el siguiente ejemplo:



Ejercicio 3.

Una empresa de turismo realizó un mapa formado por un conjunto de ciudades conectadas por rutas. Para cada ciudad la empresa de turismo determinó la cantidad de días que los turistas deben pasar en la misma para visitar todos los sitios de interés. La empresa determina que se deben pasar esa cantidad de días, ni menos ni más, ya que menos implica desaprovechar la estancia porque no se visitan todos los lugares turísticos, y más días es perder el tiempo.
Considerando que la red de ciudades se representa por un grafo, en donde las ciudades son los nodos y las rutas las aristas que las conectan, debe implementar un método en la clase Grafo que reciba una cantidad total de días de vacaciones que desea tomarse y el algoritmo determine cuáles son las ciudades que debe visitar para aprovechar exactamente todos los días. Tenga presente que las ciudades deben ser contiguas (deben tener un camino que las conecta directamente). y que el circuito (camino) puede comenzar en cualquier ciudad. También considere que para una cantidad de días pueden existir distintos caminos posibles, con lo cual es suficiente con encontrar uno cualquiera.