Programación 3 - Curso 2011
1er Modulo – 2do parcial - Viernes 3 de Junio

Ejercicio 1
Sea el siguiente algoritmo:

```
public void  sumatorias (int n) {
   int k = 1
    while (k < n) {
       for (int i = 1; i < n; i++)
              for (int j = 1; j < i; j++){
                     Operacion()
          }
       K = k *2;
       }
}
```

a.- Calcular el T(n) y el O(n). Considere Operación() de tiempo constante. **(1 punto el T(n) y 1 punto el O(n))**

Calculo del T(n):

$$c+\sum_{k=1}^{Log_2(n)}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{i}(d)\right)\right)=$$

$$=c+\sum_{k=1}^{Log_2(n)}\left(\sum_{i=1}^{n}id\right)=$$

$$=c+\sum_{k=1}^{Log_2(n)}\left(d\sum_{i=1}^{n}i\right)=$$

$$=c+\sum_{k=1}^{Log_2(n)}\left(d*\frac{n(n+1)}{2}\right)=$$

$$=c+Log_2(n)*d*\frac{n(n+1)}{2}=$$

$$=c+\frac{d}{2}*Log_2(n)*(n^2+n)=$$

$$=c+\frac{d}{2}*Log_2(n)*n^2+\frac{d}{2}*Log_2(n)*n=$$

$$=\frac{d}{2}*Log_2(n)*n^2+\frac{d}{2}*Log_2(n)*n+c=$$

Calculo del O(n):

$$T(n)=\frac{d}{2}*Log_2(n)*n^2+\frac{d}{2}*Log_2(n)*n+c\_es\_O(Log_2(n)*n^2)$$

$$\frac{d}{2}*Log_2(n)*n^2<=\frac{d}{2}Log_2(n)*n^2,\forall n_0$$

$$\frac{d}{2}*Log_2(n)*n<=\frac{d}{2}Log_2(n)*n^2,\forall n_0$$

$$c<=cLog_2(n)*n^2,\forall n_0$$

$$\frac{d}{2}*Log_2(n)*n^2+\frac{d}{2}*Log_2(n)*n+c<=\left(\frac{d}{2}+\frac{d}{2}+c\right)Log_2(n)*n^2,\forall n_0\therefore es\_O(Log_2(n)*n^2)$$

b.- Resolver la siguiente recurrencia y Calcular el O(n). **(1 punto la recurrencia y 1 punto el O(n))**

$$T(n)=\begin{cases}1 & \text{si } n = 1\\ 32\,T(n/2)+n^5 & \text{si } n \geq 2\end{cases}$$

Resolución de la recurrencia:

$$= 32T\left(\frac{n}{2}\right) + n^5 =$$

$$= 32\left(32T\left(\frac{n}{2/2}\right) + \left(\frac{n}{2}\right)^5\right) + n^5 = 32^2\left(T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2}\right)^5\right) + n^5 = 32^2 T\left(\frac{n}{2^2}\right) + 32\left(\frac{n}{2}\right)^5 + n^5 = 32^2 T\left(\frac{n}{2^2}\right) + 2n^5 =$$

$$= 32^2 T\left(\frac{n}{2^2}\right) + 2n^5 = 32^2\left(16T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2}\right)^5\right) + 2n^5 = 32^3 T\left(\frac{n}{2^3}\right) + 32^2\left(\frac{n}{2^2}\right)^5 + 2n^5 = 32^3 T\left(\frac{n}{2^3}\right) + 3n^5 =$$

$$= 32^i T\left(\frac{n}{2^i}\right) + in^5 =$$

$$\frac{n}{2^i} = 1 \rightarrow i = Log_2(n)$$

$$= 32^{Log_2(n)} T\left(\frac{n}{2^{Log_2(n)}}\right) + Log_2(n)n^5 =$$

$$= n^5 T\left(\frac{n}{n}\right) + Log_2(n)n^5 =$$

$$= n^5 + Log_2(n)n^5$$

Calculo del O(n):

$$T(n) = n^5 + Log_2(n)n^5 \_es\_O(n^5 Log_2(n))$$

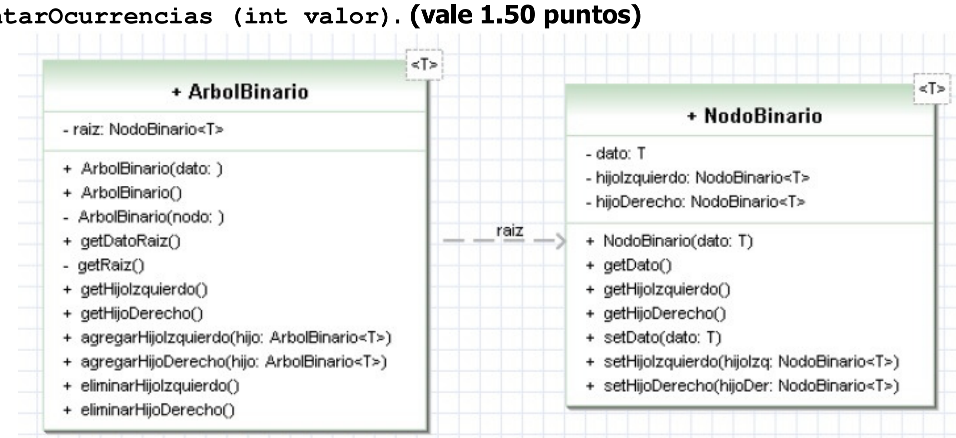
$$Log_2(n)n^5 <= n^5 Log_2(n), \forall n_0$$

$$n^5 <= n^5 Log_2(n), \forall n_0$$

$$n^5 + n^5 Log_2(n) <= 2n^5 Log_2(n), \forall n_0 \therefore es\_O(n^5 Log_2(n))$$

Ejercicio 2

Sea la siguiente definición de un árbol binario, implemente un método que cuente las veces que aparece un cierto valor: `int contarOcurrencias (int valor).` **(vale 1.50 puntos)**

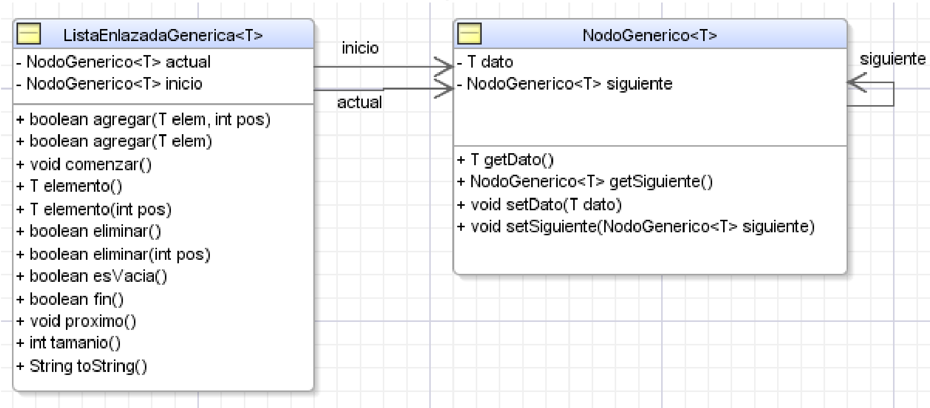


Solución:

Consideraciones: Considerar un Arbol de Integers y que el metodo contarOcurrencias recibe como parámetro un Intenger.

```
int contarOcurrencias (Integer valor){
int cantidad = 0;
    if (getDatoRaiz().equalTo(valor)) then cantidad = 1;
    cantidad = cantidad + this.getHijoIzquierdo().contarOcurrencias(valor);
    cantidad = cantidad + this.getHijoDerecho().contarOcurrencias(valor);
return cantidad
}
```

Ejercicio 3
Sean las siguientes definiciones de **`ListaGenerica<T>`** y **`NodoGenerico<T>`**:



a.- Defina usando JAVA la clase **`ArbolGeneral`** e implemente dos constructores, uno para crear un árbol vacío y otro para crearlo con un único elemento. **(0,5 puntos cada constructor)**

Solucion:
En el enunciado del parcial, brindamos la clase ListaGeneral, pero no brindamos la clase NodoGeneral, por lo cual, es necesario definir NodoGeneral de la siguiente forma:

```
public class NodoGeneral<T> {
    private T dato;
    private ListaGenerica<NodoGeneral<T>> listaHijos;
    public NodoGeneral(T dato){
        this.dato = dato;
        listaHijos = new ListaEnlazadaGenerica<NodoGeneral<T>>();
    }
```

Y luego hacer:

```
public class ArbolGeneral<T> {
        private NodoGeneral<T> raiz;

        public ArbolGeneral() {
            this.raiz = null;}

        public ArbolGeneral(T dato) {
            this.raiz = new NodoGeneral<T>(dato);}
```

b.- Implemente la operación **`preorden()`**. **(Vale 1 punto)**

Solucion:
Se basa en utilizar la operacion getHijos() del arbol y luego recorrer la lista con las operaciones que se muestran en el enunciado.

```
public void preorden() {
    // hacer algo con this.getDatoRaiz()
    ListaGenerica<ArbolGeneral<T>> hijos = this.getHijos();
    hijos.comenzar();
    while (!hijos.fin()) {
         hijos.elemento().preorden()
         hijos.proximo();
    }
}
```

Ejercicio 4
a.- Determine el tiempo requerido por un algoritmo para resolver un problema de tamaño n = 10000. Asuma que el algoritmo requiere f(n) operaciones y procesa 100 operaciones por segundo. Marque las celdas correctas en la siguiente tabla: **(vale 1 punto)**

f(n)	**< 0.01**	**0.01 – 0.5 seg**	**0.5 – 30 seg**	**1 – 3**	**5 – 7**	**1 – 24**	**> 10**

	Seg			**min**	**min**	**h**	**días**
$\log_{10}$ n							
√n							
n							
n $\log_{10}$ n							
n^2							

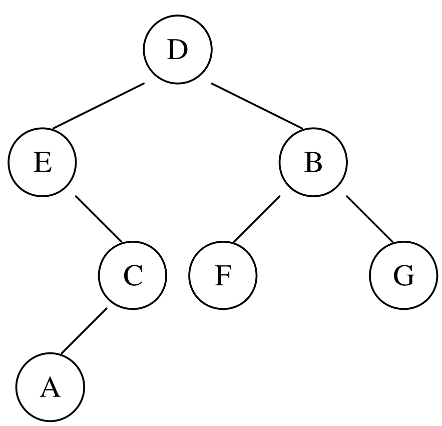
Solucion:
n = 10000
100 operaciones por segundo.

F(n)	N=10000	/100 = segundos	
Log10(n)	4	0.04	0.01-0.5 segund
Raiz (n)	100	1	0.5-30 segundos
N	10000	100	1-3 minutos
Nlog10(n)	40000	400	5-7 minutos
N2	100.000.000	1.000.000	>10 dias

F(n)	<0.01 segundos		0.5-30 segundos	1-3 minutos	5-7 minutos		>10 dias
Log10(n)		X					
Raiz (n)			X				
N				X			
Nlog10(n)					X		
N2							X

b.- Suponga que sabemos que un árbol binario, en el que cada nodo está etiquetado con una letra mayúscula, tiene el recorrido inorden: EACDFBG y el recorrido postorden: ACEFGBD. Dibuje el árbol correspondiente. El árbol puede no ser lleno, de manera que algunos nodos pueden tener sólo un hijo. En este caso muestre claramente si el único hijo es izquierdo o derecho. **(Vale 1 punto)**

Solucion:
El recorrido inorden es EACDFBG y el posorden es ACEFGBD.
Del recorrido posorden podemos deducir que D es la raiz del arbol. Si D es la raiz del arbol, del recorrido inorden podemos deducir que EAC es el arbol izquierdo y FBG. El recorrido posorden del arbol izquierdo es ACE y el recorrido posorden del arbol derecho es FGB. Para el arbol izquierdo, como el inorden es EAC, se esperarlo que A sea la raiz, y los hijos izquierdo y derecho sean E y C respectivamente. Sin embargo, para ello, el posorden debe ser ECA, que no lo es. Por el posorden sabemos que A esta en el nivel mas bajo, le sigue C y arriba E. Por el inorden, E debe ser la raiz, C el derecho, y A el izquierdo de C. Luego, el subárbol derecho es trivial, B la raiz, y los hijos F y G.



c.- Indique cuál de las siguientes es la definición de grado de un árbol **(vale 0.5 puntos)**

(a) El máximo número de nodos en uno de sus niveles
(b) El máximo número de hijos de un nodo
(c) La longitud máxima desde la raíz a una hoja
(d) Ninguna de las anteriores

Solucion:

(b), el máximo grado de hijos de un nodo.