Programación 3 - Curso 2011
1er Modulo – Viernes 20 de Mayo

Ejercicio 1
Sea el siguiente algoritmo:

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

a.- Calcular el T(n) y el O(n). Considere Operación de tiempo constante. **(vale 2 puntos, 1 punto el T(n) y 1 punto el O(n))**

Calculo del T(n):

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

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

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

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

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

$$=cn^2-c\sum_{i=1}^{n}i+cn+dn^2Log_2(n)-dLog_2(n)\sum_{i=1}^{n}i+dnLog_2(n)=$$

$$=cn^2-c\frac{n(n+1)}{2}+cn+dn^2Log_2(n)-dLog_2(n)\frac{n(n+1)}{2}+dnLog_2(n)=$$

$$=cn^2-\frac{c}{2}n^2-\frac{c}{2}n+cn+dn^2Log_2(n)-\frac{d}{2}Log_2(n)n^2-\frac{d}{2}Log_2(n)n+dnLog_2(n)=$$

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

Calculo del O(n):

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

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

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

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

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

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

b.- Resolver la siguiente recurrencia y Calcular el O(n). **(vale 2 puntos, 1 punto la recurrencia y 1 punto el O(n))**

$$T(n) = \begin{cases} 1 & \text{si } n = 1 \\ 16\,T(n/2) + n^4 & \text{si } n \geq 2 \end{cases}$$

Resolución de la recurrencia:

$$= 16T\left(\frac{n}{2}\right) + n^4 =$$

$$= 16\left(16T\left(\frac{n}{2/2}\right) + \left(\frac{n}{2}\right)^4\right) + n^4 = 16^2\left(T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2}\right)^4\right) + n^4 = 16^2 T\left(\frac{n}{2^2}\right) + 16\left(\frac{n}{2}\right)^4 + n^4 = 16^2 T\left(\frac{n}{2^2}\right) + 2n^4 =$$

$$= 16^2 T\left(\frac{n}{2^2}\right) + 2n^4 = 16^2\left(16T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2}\right)^4\right) + 2n^4 = 16^3 T\left(\frac{n}{2^3}\right) + 16^2\left(\frac{n}{2^2}\right)^4 + 2n^4 = 16^3 T\left(\frac{n}{2^3}\right) + 3n^4 =$$

$$= 16^i T\left(\frac{n}{2^i}\right) + in^4 =$$

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

$$= 16^{Log_2(n)} T\left(\frac{n}{2^{Log_2(n)}}\right) + Log_2(n)n^4 =$$

$$= n^4 T\left(\frac{n}{n}\right) + Log_2(n)n^4 =$$

$$= n^4 + Log_2(n)n^4$$

Calculo del O(n):

$$T(n) = n^4 + Log_2(n)n^4 \_ es \_ O(n^4 Log_2(n))$$

$$Log_2(n)n^4 <= n^4 Log_2(n), \forall n_0$$

$$n^4 <= n^4 Log_2(n), \forall n_0$$

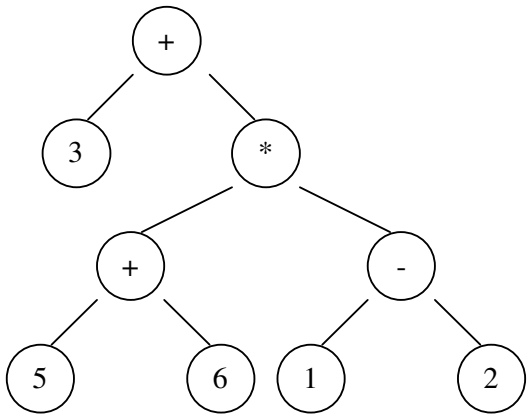
$$n^4 + n^4 Log_2(n) <= 2n^4 Log_2(n), \forall n_0 \therefore es \_ O(n^4 Log_2(n))$$

Ejercicio 2

Sea la siguiente expresión en formato prefijo: **+ 3 * + 5 6 – 1 2**

a.- Representarla en el árbol binario correspondiente. **(vale 0,5 puntos)**

Solución:

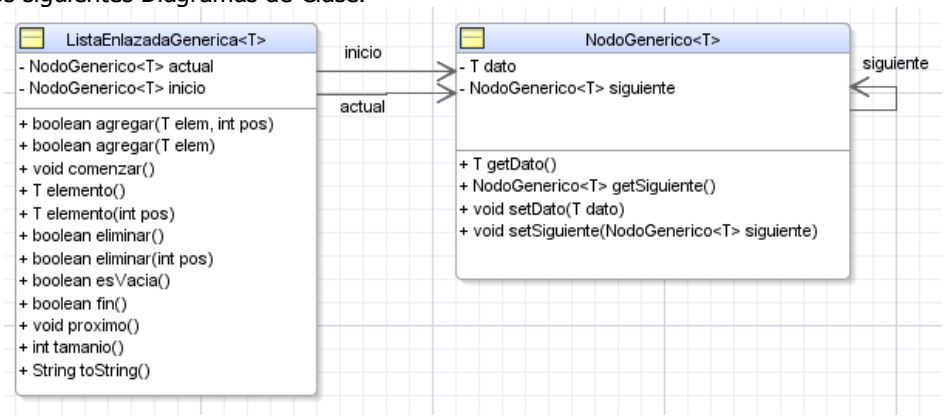


b.- A partir del árbol, obtener los recorridos preorden, inorden y posorden. **(vale 0,5 puntos)**

Solución:
Preorden: + 3 * + 5 6 – 1 2
Indorden: 3 + 5 + 6 * 1 – 2
Posorden: 3 5 6 + 1 2 - * +

Ejercicio 3
Una cola circular es una estructura en la cual, además de poder agregarse y eliminarse elementos como en una cola común, existe la operación rotación, la cual consiste en devolver el tope y encolarlo a la vez.

Para esta implementación cuenta con las clases **`ListaGenerica<T>`** y **`NodoGenerico<T>`** que se muestran en los siguientes Diagramas de Clase.



a.- Defina usando JAVA la clase **`ColaCircular`** e implemente dos constructores, uno para crear una cola vacía y otro para crearla con un único elemento. **(vale 1 punto, 0,5 puntos cada operación)**

b.- Implemente las operaciones **`poner(T elem)`**, **`sacar()`** y **`rotar()`**.**(vale 1 punto, 0,33 puntos cada operación)**

Solucion:
Se podrían realizar distintos diseños. Seria posible implementar una cola circular a partir de una lista, o se podría definir la cola circular a partir de una cola. A continuación mostramos una solución donde se implementa la cola circular extiendo la clase cola, la cual a su vez se implementa a partir de una lista.

```
public class ColaCircular<T> extends ColaGenerica<T> {
	public ColaCircular() {
		super();
	}
	public ColaCircular(T dato) {
		super();
		this.getDatos().agregar(dato);
	}
	public void rotar(){
		T dato  = this.sacar();
		this.poner(dato);
	}
}


public class ColaGenerica<T> {
	private ListaGenerica<T> datos;

	public ListaGenerica<T> getDatos() {
		return datos;
	}

	public ColaGenerica() {
		this.datos = new ListaEnlazadaGenerica<T>();
	}

	public void poner(T dato) {
```

```
            datos.agregar(dato, datos.tamanio());
        }

        public T sacar() {
            T x = datos.elemento(0);
            datos.eliminar(0);
            return x;
        }

        public T tope() {
            return datos.elemento(0);
        }

        public boolean esVacia() {
            return datos.tamanio() == 0;
        }
}
```

c.- Puede afirmar que la operación **`rotar()`** es de orden lineal? Justifique. **(vale 1 punto)**

Solucion:
Dado que la operación rotar, invoca a las operaciones sacar (de orden constante) y la operación poner (de orden lineal por recorrer toda la lista hasta llegar al final de la misma), podemos concluir que es de orden lineal.

Ejercicio 4
a.- Mencione la cantidad mínima y máxima de nodos en un árbol binario completo de altura h. Justifique su respuesta **(vale 1 punto)**

Solución:
Un árbol completo de altura h es un árbol lleno de altura h – 1, en donde el nivel h se va ocupando de izquierda a derecha. La siguiente tabla muestra la delación:

H	N
0	1
1	3
2	7
I	$2^{i+1}-1$

Por lo tanto, un arbol completo de altura h de cantidad minima, tendra el nivel h-1 lleno y un nodo mas, lo cual se traduce como: $2^{h-1+1}-1+1=2^h$.
Por su parte, la cantidad máxima de un arbol completo se corresponde con la cantidad de nodos de un arbol lleno: $2^{h-1+1}-1$.

b.- Plantee la función de Tiempo de ejecución del recorrido PreOrden para los Árboles Generales. Plantear solo la recurrencia? **(vale 1 punto)**

Solución:

Considerando k el grado, la recurrencia utilizando n es:

$$T(n) = \begin{cases} 1 & \text{si } n = 1 \\ C+kT(n-1/k) & \text{si } n \geq 2 \end{cases}$$

Considerando k el grado, la recurrencia utilizando h es:

$$T(h) = \begin{cases} 1 & \text{si } n = 1 \\ C+kT(h-1) & \text{si } n \geq 2 \end{cases}$$

c.- Formalmente se dice que f(n) es O(g(n)) si $\exists$ c, $\exists$ n0, tal que $\forall$ n $\geq$ n0, se cumple f(n) $\leq$ c g(n). ¿Qué valores de **c** y $\mathbf{n_0}$ elige en esta definición para mostrar que $2n^2 - 3n$ es $O(n^2)$? **(vale 0,5 puntos)**

(a) $n_0 = 2$, $c = 1$
(b) $n_0 = 3$, $c = 2$
(c) (a) y (b)

(d) Ninguna de las anteriores

Solución:

Si consideramos: $n_0 = 2$, $c = 1$

N	T(n)	O(n)
	$2*n^2 - 3*n$	$C* n^2$
2	$2*2^2 - 3*2 = 8 - 6 = 2$	$1* 2^2 = 4$
3	$2*3^2 - 3*3 = 18 - 9 = 9$	$1* 3^2 = 9$
4	$2*4^2 - 3*4 = 32 - 12 = 20$	$1* 4^2 = 16$

Vemos que para n=4 no se verifica la desigualdad. Si consideramos: $n_0 = 3$, $c = 2$

N	T(n)	O(n)
	$2*n^2 - 3*n$	$C* n^2$
3	$2*3^2 - 3*3 = 18 - 9 = 9$	$2* 3^2 = 18$
4	$2*4^2 - 3*4 = 32 - 12 = 20$	$2* 4^2 = 32$
5	$2*5^2 - 3*5 = 50 - 15 = 35$	$2* 5^2 = 50$

Podemos ver que O(n) siempre acotará a T(n), por lo cual, la respuesta correcta es (b)