

# Prueba de Resiliencia

## Técnicas y Metodologías de Programación Avanzada – 05-06-2025

Nombre		RUT	
Asignatura	( ) Sebastián Arce		

### Antecedentes generales:

Puntaje total de la prueba/Puntos para nota aprobatoria (4.0)	50 puntos 30 puntos	Puntaje Obtenido	
Duración de la prueba	1,5 horas	Nota final	
Resultados de Aprendizaje a evaluar	Aplicar estrategias avanzadas recursivas y de fuerza bruta para resolver problemas de ingeniería		
Fecha de entrega de resultados			

### Instrucciones:

1. Esta evaluación tiene 3 páginas (incluyendo la portada) y 4 preguntas. Compruebe que dispone de todas las páginas. Responda a las preguntas en los espacios previstos para ello en las hojas de preguntas. Asegúrese de marcar apropiadamente sus respuestas.
2. Durante la prueba no se puede utilizar: teléfono móvil, calculadora, apuntes y/o computador. Está prohibido intentar conectarse a internet de cualquier manera. Si es sorprendido obtendrá la calificación mínima. Tampoco puede utilizar dispositivos de almacenamiento externos o cualquier otro dispositivo (relojes inteligentes, etc.).
3. Lea la prueba completamente DOS veces antes de hacer cualquier pregunta
4. Una prueba respondida correctamente en un 60%, de acuerdo con las ponderaciones asignadas, corresponde a una nota 4,0.
5. Solamente se pueden realizar preguntas durante los primeros 10 minutos de la prueba. Solo se responderán preguntas respecto a los enunciados a viva voz.
6. La prueba es individual, cualquier sospecha de copia será calificada con la nota mínima y el caso será remitido al comité de ética.
7. En su espacio personal no debe haber nada más que hojas de papel en blanco, lápiz, goma y/o calculadora.
8. El resto de sus implementos debe guardarlos dentro de su mochila/bolso y ésta debe posicionarse al frente debajo de la pizarra. Si leyó hasta este punto, felicidades, para saber que lo hizo dibuje una estrella al final de esta página.

Acepto las condiciones firmando: \_\_\_\_\_

### Problema 1 (10 puntos)

Haz un programa en Java que lea un **número entero positivo** desde teclado y determine, usando **recursión**, si el número es **capicúa**.

(Un número capicúa es el que se lee igual de izquierda a derecha que de derecha a izquierda).

Ejemplos:

- 121 → es capicúa
- 3443 → es capicúa
- 123 → no es capicúa

No puede transformar el valor **entero** a otro tipo de variable. Implementar la función principal y la función recursiva.

Nombre de los métodos a implementar:

- `public static void main()`
- `public int capicua()`

### Problema 2 (15 puntos)

Utilizando **dividir para conquistar**, dada una **matriz cuadrada de números enteros**, implemente un algoritmo recursivo que calcule el **producto de los elementos de la diagonal secundaria que sean divisibles por 3**.

Si ningún elemento cumple la condición, debe imprimir **1** como resultado.

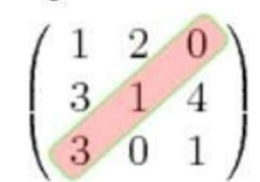
- Tomar como guía las funciones ya creadas en la siguiente hoja.

Implemente la función correspondiente al enunciado.

Nombre de los métodos a implementar:

- `public int multiplicarDivisiblesPor3()`

Diagonal Secundaria



### Problema 3 (15 puntos)

Considere la organización de datos de personas en un **árbol binario de búsqueda**, ordenado por ID.

Cada nodo contiene un ID y rendimiento.

Se solicita que implemente las funciones utilizando **recursión**:

- Se pide implementar un método recursivo en Java que, dados dos IDs, encuentre su **ancestro común más cercano**, y luego calcule la **suma de rendimiento de todos los nodos que pertenecen al subárbol enraizado en dicho ancestro**.

Nombre de las funciones a implementar:

- `public int sumaRendimientoDesdeAncestro(String id1, String id2)`

Tomar como guía las funciones ya creadas en la siguiente hoja.

### Problema 4 (10 puntos)

Rutee claramente el siguiente código Java, e indique lo que imprime.

```
public class Ruteo {  
  
    public static void main(String[] args) {  
  
        int k = 6;  
  
        int valor = metodo(k, 1);  
  
        System.out.println("valor " + valor);  
  
    }  
}
```

```
public static int metodo(int m, int n) {  
    if (m <= 0) {  
        return 3 * (m + n);  
    } else {  
        return (3 + m) * metodo(m - 1, n);  
    }  
}  
}  
}
```

**Problema 2, código ya creado.**

```
public static void main(String[] args) {  
  
    int k = 3;  
    int N = (int) Math.pow(2, k);  
    int[][] matriz = generarMatriz(N);  
  
    System.out.println("Matriz:");  
  
    int resultado = dividirParaConquistar(matriz, 0, N - 1);  
  
    System.out.println("Producto de los elementos divisibles por 3 en la diagonal  
secundaria: " + resultado);  
}
```

**Problema 3, código ya creado.**

```
// Clase NodoEmpleado  
  
class NodoEmpleado {  
  
    String id;  
    int rendimiento;  
    NodoEmpleado left, right;  
  
    public NodoEmpleado(String id, int rendimiento) {  
  
        this.id = id;  
        this.rendimiento = rendimiento;  
        this.left = null;  
        this.right = null;  
    }  
}
```

```
class ArbolEmpleado {  
  
    NodoEmpleado root;  
  
    public ArbolEmpleado() {  
        root = null;  
    }  
  
    public void insertar(String id, int rendimiento) {  
        root = insertarRec(root, id, rendimiento);  
    }  
}
```