# Paradigmas de la Programación – Recuperatorio del Primer Parcial

18 de junio de 2019

**Apellido y Nombre:** _____

1. Dado el siguiente programa:

   - **[10 pt.]** Diagrame los estados por los que va pasando la pila de ejecución al ejecutarse el programa, asumiendo que el lenguaje tiene alcance estático.
   - **[10 pt.]** ¿Qué se imprimiría si el lenguaje tiene alcance estático? ¿Y si tiene alcance dinámico?
   - **[10 pt.]** ¿Qué se imprimiría si el lenguaje tiene pasaje de parámetros por valor? ¿Y si tiene pasaje de parámetros por referencia?

```
var x : int = 0;
var y : int = 1;

procedure M(x : int)
begin
    var y : int = 42;
    x := 1;
    call N(x);
end

procedure N(x : int)
begin
    var z : int = 2;
    if true begin
        var z : int = 42;
    end
    print x;
    print y;
    print z;
end

procedure Main()
begin
    call M(x);
end
```

2. **[10 pt.]** La siguiente expresión está mal tipada:

```
f (a, b) = ( b + a > 3 ) || b
```

Muestre con el árbol para inferencia de tipos dónde se encuentra el conflicto y en qué consiste. Cómo podría resolver este conflicto un lenguaje de tipado fuerte? y uno de tipado débil?

3. **[10 pt.]** Explique por qué la instrucción `go to` puede producir código no estructurado. Explique a partir de eso cuál es la diferencia esencial entre código estructurado y no estructurado, y qué impacto tiene esa diferencia en la gestión de la memoria. La instrucción `go to`, ¿puede usarse siguiendo las reglas de código estructurado?

4. **[10 pt.]** Teniendo en cuenta que la siguiente función en Javascript tiene transparencia referencial ¿qué diferencia hay entre pasar los parámetros por valor y pasarlos por referencia?

```
const sum = a => b => a + b;
console.log (sum (5) (3));
```

5. **[10 pt.]** Estas dos funciones tienen la misma semántica, pero una de ellas no es declarativa. ¿Cuál no es declarativa y por qué no lo es?

```
const container = document.getElementById('container');
const btn = document.createElement('button');
btn.className = 'btn red';
btn.onclick = function(event) {
 if (this.classList.contains('red')) {
   this.classList.remove('red');
   this.classList.add('blue');
 } else {
   this.classList.remove('blue');
   this.classList.add('red');
 }
};
container.appendChild(btn);
```

```
class Button extends React.Component{
  this.state = { color: 'red' }
  handleChange = () => {
    const color = this.state.color === 'red' ? 'blue' : 'red';
    this.setState({ color });
  }
  render() {
    return (<div>
      <button
          className=`btn ${this.state.color}`
          onClick={this.handleChange}>
      </button>
    </div>);
  }
}
```

6. **[10 pt.]** En el siguiente código en Java, cuándo se puede recolectar el objeto `Bar` creado en la línea 6? Justifique su respuesta usando el concepto de *alcanzabilidad*.

```
class Bar { }
class Test
{
    Bar doBar()
    {
        Bar b = new Bar(); /* Linea 6 */
        return b; /* Linea 7 */
    }
    public static void main (String args[])
    {
        Test t = new Test();  /* Linea 11 */
        Bar newBar = t.doBar();  /* Linea 12 */
        System.out.println("newBar");
        newBar = new Bar(); /* Linea 14 */
        System.out.println("finishing"); /* Linea 15 */
    }
}
```

7. **[10 pt.]** Diagrame los estados por los que pasa la pila de ejecución en el siguiente programa en java, enfocándose únicamente en el manejo de excepciones, sin representar las variables locales, control links, access links, retorno de función ni ninguna otra información que no sea relevante al proceso de manejo de excepciones. Explique qué pasa con los dos `catch`, y por qué.

```
class Test
{
    public static void main(String[] args)
    {
        try
        {
            int a[]= {1, 2, 3, 4};
            for (int i = 1; i <= 4; i++)
            {
                System.out.println ("a[" + i + "]=" + a[i] + "n");
            }
        }
        catch (Exception e)
        {
            System.out.println ("error = " + e);
        }
        catch (ArrayIndexOutOfBoundsException e)
        {
            System.out.println ("ArrayIndexOutOfBoundsException");
        }
    }
}
```