# Paradigmas de la Programación – Examen Final

7 de Julio de 2016

**Apellido y Nombre:** _____

1. **[5 pt.]** El siguiente es un ejemplo de "*spaghetti code*". Reescríbalo en pseudocódigo de forma que NO use saltos (GOTO), y en cambio use programación estructurada en bloques.

```
10 i = 0
20 i = i + 1
30 PRINT i; " squared = "; i * i
40 IF i >= 10 THEN GOTO 60
50 GOTO 20
60 PRINT "Program Completed."
70 END
```

2. **[10 pt.]** Muestre con un ejemplo que la siguiente gramática es ambigua, y modifíquela para que se convierta en una gramática totalmente inambigua.

```
<a> ::= <b> <c>
<b> ::= <a>
<b> ::= []
<c> ::= 'c' 'd' 'e'
<c> ::= 'c' 'd'
<c> ::= 'e'
```

3. **[15 pt.]** Diagrame una secuencia de pilas de ejecución que representen los los diferentes momentos de la ejecución del siguiente programa, mostrando cómo se apilan y desapilan los diferentes *activation records* a medida que se va ejecutando el programa. Asuma que el lenguaje de programación tiene alcance estático. Diga qué retorna f(2) si el lenguaje tiene alcance estático, y qué retornaría con alcance dinámico. Señale también los puntos de la ejecución en los que el recolector de basura puede recolectar variables y diga cuáles pueden ser recolectadas.

```
var x=1;
var y=5;
function g(z) {return x+z;}
function f(y) {
  var x = y*y;
  return g(y);
}
var x=5;
f(2)
```

4. **[10 pt.]** Escriba un programa (puede ser en pseudocódigo) en el que el resultado sea distinto si el pasaje de parámetros es por valor, por referencia o por valor-resultado. Explique qué comparte el pasaje de parámetros por valor-resultado con el pasaje por valor y qué comparte con el pasaje por referencia.

5. **[10 pt.]** El enlace dinámico (o *dynamic dispatch*) es un mecanismo de la programación orientada a objetos por el cual se selecciona con qué método o función se responderá a un determinado mensaje en tiempo de ejecución. Es necesario hacer esto en tiempo de ejecución porque es imposible saber en tiempo de compilación cuál será el tipo del mensaje recibido. En C++ el tipo de *dispatch* por defecto es estático, y para que se pueda realizar *dynamic dispatch* hay que declarar un método como `virtual`. Los métodos virtuales se implementan mediante una estructura de datos que se llama `vtable` o `virtual table`, que define la correspondencia entre mensajes y métodos para cada clase. Comente la utilidad y coste del *dynamic dispatch* en términos de overhead y flexibilidad. Establezca un paralelismo con el polimorfismo de tipos mediante un ejemplo.

6. **[10 pt.]** Supongamos que un lenguaje de programación usa el símbolo "=" como operador de unificación, ¿qué imprimiría el siguiente programa?

```
var x;
var y=1;
function f(a,b) {
  if ( a = b ) { print "si" }
  else { print "no" }
}
f(x,y)
```

7. **[10 pt.]** Este código en Erlang implementa las mismas funcionalidades que los canales de Go mediante los procesos y mensajes de Erlang. Identifique porciones del programa que tienen semántica puramente concurrente. Identifique aquellas que tienen funcionalidades específicas para pasaje de mensajes, como en el modelo de actores.

```
defmodule GoChannel do
  def make do
    spawn(&GoChannel.loop/0)
  end

  def write(channel, val) do
    send(channel, { :write, val})
  end

  def read(channel) do
    send(channel, { :read, self })

    receive do
      { :read, channel, val} -> val
    end
  end

  def loop do
    receive do
      { :read, caller } -> receive do
```

```
        { :write, val } -> send(caller, { :read, self, val }); loop
      end
    end
  end
end
```

8. **[10 pt.]** En las siguientes funciones en ML:

```
exception Excpt of int;
fun twice(f,x) = f(f(x)) handle Excpt(x) => x;
fun pred(x) = if x = 0 then raise Excpt(x) else x-1;
fun dumb(x) = raise Excpt(x);
fun smart(x) = 1 + pred(x) handle Excpt(x) => 1;
```

Cuál es el resultado de evaluar cada una de las siguientes expresiones?

a) `twice(pred,1)`

b) `twice(dumb,1)`

c) `twice(smart,1)`

Explique qué excepción se levanta en cada caso y dónde se levanta. Ayúdese de los respectivos diagramas de pilas de ejecución.

9. **[5 pt.]** En Perl las variables pueden tener alcance estático (léxico) o dinámico, usando distintas palabras clave en el momento de su asignación. En el siguiente programa se ve cómo las palabras clave "`local`" y "`my`" tienen distintos efectos en la forma cómo las variables adquieren su valor. Diga cuál de estas dos palabras claves se usa para que el valor de la variable se asigne con alcance estática y cuál con alcance dinámico, y explique por qué.

```
sub visible {
    print "la variable tiene el valor $var\n";
}
sub uno {
    local $var = 'valor uno';
    visible();
}
sub dos {
    my $var = 'valor dos';
    visible();
}

$var = 'global';
visible();          # imprime "global"
uno();              # imprime "valor uno"
dos();              # imprime "global"
```

10. **[15 pt.]** Escriba un programa (puede ser en pseudocódigo) en el que una variable sea el medio por el cual se propagan efectos secundarios más allá del alcance de una función. Relacione este fenómeno con los paradigmas funcional e imperativo, incluyendo en su explicación los conceptos de determinismo en las componentes de software. Explique como este comportamiento que puede originar tantos comportamientos inesperados puede llegar a utilizarse de forma ventajosa en un esquema productor-consumidor.

# Ejercicios para libres

1. **[-10 pt.]** Java tiene una construcción lingüística llamada "`try... catch... finally...`" cuya semántica consiste en que el bloque de código bajo el alcance de `finally` se ejecuta siempre, tanto si se lanza una excepción bajo el alcance de `try` como si no se lanzó. Si se lanza una excepción, la excepción funciona exactamente igual que si no existiera el bloque "`finally`" (aunque el bloque `finally` se ejecuta en cualquier caso), si se puede capturar mediante el `catch` se captura, y si no sigue buscando hasta que encuentra un `catch` que la pueda capturar. Sabiendo esto, explique qué se imprime si ejecutamos los siguientes tres programas y por qué.

```
class Ejemplo1 {
   public static void main(String args[]){
     try{
        System.out.println("primera sentencia del bloque try");
        int num=45/0;
        System.out.println(num);
     }
     catch(ArrayIndexOutOfBoundsException e){
        System.out.println("ArrayIndexOutOfBoundsException");
     }
     finally{
        System.out.println("bloque finally");
     }
     System.out.println("fuera del bloque try-catch-finally");
   }
}
```

```
class Ejemplo2 {
   public static void main(String args[]){
      try{
         System.out.println("primera sentencia del bloque try");
         int num=45/0;
         System.out.println(num);
      }
      catch(ArithmeticException e){
         System.out.println("ArithmeticException");
      }
      finally{
         System.out.println("bloque finally");
      }
      System.out.println("fuera del bloque try-catch-finally");
   }
}
```

```
class Ejemplo3 {
  public static void main(String args[]){
    try{
       System.out.println("primera sentencia del bloque try");
       int num=45/3;
       System.out.println(num);
    }
```

```
    catch(ArrayIndexOutOfBoundsException e){
        System.out.println("ArrayIndexOutOfBoundsException");
    }
    finally{
        System.out.println("bloque finally");
    }
    System.out.println("fuera del bloque try-catch-finally");
  }
}
```

2. **[-5 pt.]** En el siguiente código en Ruby, describa la visibilidad de la variable `cuenta`.

```
class Ser
    @@cuenta = 0

    def initialize
        @@cuenta += 1
        puts "creamos un ser"
    end
    def muestra_cuenta
        "Hay #{@@cuenta} seres"
    end
end
class Humano < Ser
   def initialize
       super
       puts "creamos un humano"
   end
end
class Animal < Ser
   def initialize
       super
       puts "creamos un animal"
   end
end
class Perro < Animal
   def initialize
       super
       puts "creamos un perro"
   end
end

Humano.new
d = Perro.new
puts d.muestra_cuenta
```

3. **[-5 pt.]** Si el resultado del siguiente programa es 19, qué tipo de alcance tiene el lenguaje de programación en el que está escrito, estático o dinámico?

```
val x = 4;
    fun f(y) = x*y;
    fun g(x) = let
        f(3) + x;
    g(7);
```

4. **[-5 pt.]** Identifique en el siguiente código en C++ un problema con la herencia del miembro `meow`.

```
class Felino {
public:
 void meow() = 0;
};

class Gato : public Felino {
public:
 void meow() { std::cout << "miau\n"; }
};

class Tigre : public Felino {
public:
 void meow() { std::cout << "ROARRRRRR\n"; }
};

class Ocelote : public Felino {
public:
 void meow() { std::cout << "roarrrrr\n"; }
};
```