Paradigmas de la Programación – Examen Final

3 de Julio de 2015

Apellido y Nombre: _____

1. **[20 pt.]** Estos dos programas, en dos lenguajes distintos, tienen la misma semántica. ¿Cuáles son las diferencias entre estos dos lenguajes? Refiérase a los elementos de los programas que ejemplifican las diferencias entre estos dos lenguajes. Cuando sea relevante, argumente las ventajas y desventajas de la forma de expresión de cada lenguaje.

```
interface Iterable<T>
  def each(&block : T ->)
end

interface Addable<T>
  def +(other : T)
end

def sum(values : Iterable<T>) where T : Addable<T>
  count = 0
  values.each do |value|
    count += value
  end
  count
end
```

```
def sum(values)
  count = 0
  values.each do |value|
    count += value
  end
  count
end
```

2. **[20 pt.]** Describa las diferencias semánticas en el siguiente programa si se implementa en un lenguaje con alcance estático o con alcance dinámico. Si lo requiere, puede ayudarse de una secuencia de pilas de ejecución.

```
gcd' 0 y = y
gcd' x y = gcd' (y `mod` x) x
myGCD x y | x < 0      = myGCD (-x) y
          | y < 0      = myGCD x (-y)
          | y < x      = gcd' y x
          | otherwise = gcd' x y
```

3. **[10 pt.]** En el siguiente programa:

```
begin
  integer a, b, c;
  procedure p(x: integer, y: integer, z: integer);
    begin
      c := 5;
      x := z;
      c := 4;
      y := z+a
    end;
  a := 2;
  b := 4;
  c := 1;
  p(a, b, a+c);
  print(b);
end;
```

Qué dos valores se imprimen si `k` se pasa...

a) por valor?

b) por valor-resultado?

c) por referencia?

4. **[10 pt.]** Calcule el tipo de datos de la siguiente función en ML. Provea el árbol sintáctico de la función y aplique de forma explícita el algoritmo de inferencia de tipos, ya sea sobre el árbol mismo o como sistema de ecuaciones.

```
fun a(f,x,y) = f(x*2+3) andalso y
```

5. **[10 pt.]** Estos dos programas tienen una semántica aproximadamente equivalente. ¿Cuál de los dos es orientado a objetos? ¿El que no es orientado a objetos, en qué paradigma podría clasificarse? Describa informalmente la semántica de estos programas. Describa muy brevemente la estructura de cada uno de estos programas, marcando las diferencias entre ellos.

```
class Greeter
  def initialize(name)
    @name = name.capitalize
  end

  def salute
    puts "Hello #{@name}!"
  end
end

g = Greeter.new("world")
g.salute
```

```
"Hello World".each_char do |char|
  print char
end
print '\n'
```

6. **[10 pt.]** En el siguiente código, identifique las expresiones que expresan semántica exclusiva de un programa concurrente y describa su semántica informalmente.

```
class Ejemplo extends RecursiveTask<Integer> {
  final int n;
  Ejemplo(int n) { this.n = n; }
  Integer compute() {
    if (n <= 1)
       return n;
    Ejemplo f1 = new Ejemplo(n - 1);
    f1.fork();
    Ejemplo f2 = new Ejemplo(n - 2);
    return f2.compute() + f1.join();
  }
}
```

7. **[20 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

Ejercicios para libres

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. **[-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);
```

3. **[-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"; }
};
```

4. **[-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
```