Paradigmas de la Programación – Examen Final

19 de Diciembre de 2016

Apellido y Nombre: _____

1. **[5 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(x,y) = (x > 2) orelse (y < 10)
```

2. **[5 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'
```

3. **[20 pt.]** Diagrame la pila de ejecución del siguiente programa en Bash, incluyendo la estructura de control links y access links, asumiendo que el lenguaje de programación tiene alcance estático.

```
x=1
function g () { echo $x ; x=2 ; }
function f () { local x=3 ; g ; }
f # does this print 1, or 3?
echo $x
```

4. **[10 pt.]** El programa anterior está escrito en Bash, que en la realidad tiene alcance dinámico. Justifique por qué puede ser que Bash se haya diseñado con lenguaje dinámico. Qué imprimirá este programa, ahora que sabemos que tiene alcance dinámico?

5. **[10 pt.]** Si quiero un control fuerte de la seguridad de una aplicación, debo renunciar a usar un lenguaje de scripting como por ejemplo JavaScript? Dé por lo menos un argumento que muestre las vulnerabilidades de un lenguaje de scripting y por lo menos dos argumentos o estrategias con las que podría tratar de reducir los riesgos de un lenguaje de scripting. Argumente qué ventajas aportan las características del lenguaje que implican vulnerabilidades.

6. **[10 pt.]** Explique por qué se eliminó la herencia múltiple en lenguajes creados posteriormente a C++, y qué características tienen los mecanismos alternativos. Ayudándose del siguiente texto, ejemplifique su explicación mediante las características específicas de los `mixins`, y extrapole a las `interfaces` de Java (o a los `traits` de PHP o Scala o a alguna característica similar que usted conozca), notando semejanzas y diferencias. ¿Qué problemas tratan de evitar estos mecanismos? ¿Qué capacidades expresivas tienen?

> *A mixin can basically be thought of as a set of code that can be added to one or more classes to add additional capabilities without using inheritance. In Ruby, a mixin is code wrapped up in a module that a class can include or extend [...] Modules can either be included or extended depending on whether the methods will be added as instance methods or class methods, respectively.*

7. **[10 pt.]** Señale cuáles de las siguientes expresiones son `true` en Prolog, es decir, en qué casos se encuentra una unificación exitosa para la expresión, y cómo se pueden instanciar los valores de las variables en las unificaciones exitosas.

   a) `f(X,Y) = f(P,P)`

   b) `a(X,c(d,X)) = a(2,c(d,Y))`

   c) `a(X,c(d,X)) = a(X,c(d,Y))`

   d) `a(X,c(d,X)) = a(2,c(X,Y))`

   e) `f(foo,L) = f(A1,A1)`

   f) `f(a) =  a`

   g) `f(X,a) = f(a,X)`

8. **[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.

9. **[10 pt.]** La siguiente función de Python añade un elemento a una lista dentro de un diccionario de listas. Modifíquelo usando excepciones para manejar un posible error de llaves (`KeyError`) si la lista con el nombre no existe todavía en el diccionario, en lugar de comprobar por adelantado si ya existe. Incluya una cláusula `finally` (que se ejecutará independientemente de si se lanza una excepción o no).

```
def aniadir_a_lista_en_diccionario(diccionario, nombrelista, elemento):
    if nombrelista in diccionario:
        l = diccionario[nombrelista]
        print("%s ya tiene %d elementos." % (nombrelista, len(l)))
    else:
        diccionario[nombrelista] = []
        print("Creamos %s." % nombrelista)

    diccionario[nombrelista].append(elemento)

    print("Aniadimos %s a %s." % (elemento, nombrelista))
```

10. **[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
```

11. **[10 pt.]** En el siguiente código en Scheme:

```
(define incr ())
(define get ())

(let ((n 0))
   (set! incr (lambda (i) (set! n (+ n i))))
   (set! get (lambda () n)))
```

Las variables `incr` y `get` son globales, y la variable `n` se comparte entre ambas. Explique lo que hace este programa usando los conceptos de *efectos secundarios* y *variable compartida*, y explique utilidades y problemas del estado explícito en programación.

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.]** Dada la siguiente base de conocimiento en Prolog, explique cómo sería la sucesión de objetivos que hay que satisfacer para contestar la consulta `primo(X,juan).`

```
progenitor(esteban,juan).
progenitor(ana,juan).
progenitor(esteban,teresa).
progenitor(ana,teresa).
progenitor(marisa,noelia).
progenitor(felix,noelia).
progenitor(marisa,tristan).
progenitor(victor,tristan).
hermano(esteban,victor).
primo(X,Y):- tio(Z,X), progenitor(Z,Y).
tio(X,Y):- hermano(X,Z), progenitor(Z,Y).
```

3. **[-5 pt.]** En el siguiente programa,

```
begin
  integer n;
  procedure p(k: integer);
    begin
      k := k+2;
      print(n);
      n := n+(2*k);
    end;
  n := 4;
  p(n);
  print(n);
end;
```

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

a) por valor?

b) por valor-resultado?

c) por referencia?

4. **[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
```