Paradigmas de la Programación – Examen Final

6 de Agosto de 2015

Apellido y Nombre: _____

1. **[20 pt.]** Escriba el siguiente programa excepciones en lugar del manejo de errores mediante estructuras de control.
   Nota: no es necesario que el programa compile, puede usar pseudocódigo siempre que su semántica sea ambigua.

```
codigoErrorType readFile {
    initialize codigoError = 0;

    // abrir el archivo;
    if (elArchivoEstaAbierto) {
        // determinar la longitud del archivo;
        if (obtenemosLongitudArchivo) {
            // alojar esa cantidad de memoria;
            if (tenemosSuficienteMemoria) {
                // leemos el archivo a memoria;
                if (falloLectura) {
                    codigoError = -1;
                }
            } else {
                codigoError = -2;
            }
        } else {
            codigoError = -3;
        }
        // cerrar el archivo;
        if (elArchivoNoSeCerro && codigoError == 0) {
            codigoError = -4;
        } else {
            codigoError = codigoError and -4;
        }
    } else {
        codigoError = -5;
    }
    return codigoError;
}
```

2. **[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.

```
fun aplicar(f,x) = f(x)
```

3. **[20 pt.]** Escriba un programa (puede ser en pseudocódigo) que de diferentes resultados si se usa pasaje de parámetros por valor o por referencia.

4. **[20 pt.]** Indique si el siguiente programa en C++ se podría escribir en Java con una semántica equivalente. Si no se puede, indique por qué. Si se puede, indique con qué recursos del lenguaje se puede.

```
class Persona
{
private:
    std::string m_strNombre;
    int m_nEdad;
    bool m_bEsVaron;

public:
    Persona(std::string strNombre, int nEdad, bool bEsVaron)
        : m_strNombre(strNombre), m_nEdad(nEdad), m_bEsVaron(bEsVaron)
    {
    }

    std::string GetNombre() { return m_strNombre; }
    int GetEdad() { return m_nEdad; }
    bool EsVaron() { return m_bEsVaron; }
};

class Empleado
{
private:
    std::string m_strEmpleador;
    double m_dSalario;

public:
    Empleado(std::string strEmpleador, double dSalario)
        : m_strEmpleador(strEmpleador), m_dSalario(dSalario)
    {
    }

    std::string GetEmpleador() { return m_strEmpleador; }
    double GetSalario() { return m_dSalario; }
};

class Profesor: public Persona, public Empleado
{
private:
     int m_nDictaGrado;

public:
    Profesor(std::string strNombre, int nEdad, bool bEsVaron,
    std::string strEmpleador, double dSalario, int nDictaGrado)
        : Persona(strNombre, nEdad, bEsVaron),
        Empleado(strEmpleador, dSalario),
        m_nDictaGrado(nDictaGrado)
    {
    }
};
```

5. **[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
```

Este programa 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?

6. **[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)`

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
```