Paradigmas de la Programación – Examen Final

6 de Agosto de 2015

Apellido y Nombre: _____

1. **[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)
   ```

2. **[10 pt.]** En el siguiente programa,

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

   Qué valores se imprimen si el lenguaje tiene pasaje de parámetros...

   a) por valor?

   b) por valor-resultado?

   c) por referencia?

3. **[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) `pred(X,var) = pred(var,X)`

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

   c) `pred(X,Y) = pred(P,P)`

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

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

   f) `pred(var) =  var`

   g) `bin(X,c(d,X)) = bin(X,c(d,Y))`

4. **[10 pt.]** En el siguiente programa en C++ indique si hay *name clashes*. Si los hay, identifíquelos y mencione alguna política para solucionarlos. Si no los hay, introdúzcalos y añada los mecanismos para solucionarlos (no es necesario que sea el mecanismo que efectivamente implementa C++, puede ser otra política). ¿En qué contexto se dan los *name clashes*?

```
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.]** Explique por qué se dice que las excepciones tienen alcance dinámico. Use como ejemplo el siguiente código, describiendo los valores que tiene la variable `x` en el momento de ser atrapada (`handle`) y comparando esos valores con los valores que tendría la variable en el bloque de código donde se encuentra el `handle` si su alcance fuera estático.

```
fun twice(f,x) = f(f(x)) handle Excpt(x) => x;
fun pred(x) = if x = 0 then raise Excpt(x); 5; else x-1;
fun dumb(x) = raise Excpt(x); 3;
fun smart(x) = 1 + pred(x); 4 handle Excpt(x) => 1;
```

6. **[10 pt.]** En java existen dos formas de crear un nuevo *thread* explícitamente: creando una subclase de la clase `Thread` o bien implementando la interfaz `Runnable`. En ambos casos el nuevo *thread* debe implementar el método `run`, que contendrá el código ejecutado por el *thread*. Cuál de estas dos opciones crea un objeto más versátil y por qué?

7. **[10 pt.]** Diga si la sentencia

   ```
   ( ( 5 ) )
   ```

   pertenece al lenguaje descripto por la siguiente gramática:

   ```
   <s_exp> ::= <atomic_symbol> | <natural> | "(" <s_exp> "." <s_exp> ")"
   <atomic_symbol> ::= <letter><atom_part> | ":" <atom_part>
   <natural> ::= {<number>}+
   <atom_part> ::= <empty> | <letter><atom_part> | <number><atom_part>
   <letter> ::= "a" | "b" | ... | "z"
   <number> ::= "1" | "2" | ... | "9"
   <empty> ::= ""
   ```

   Si pertenece, muestre la secuencia de reglas que lo generan, si no pertenece, agregue las reglas necesarias para que la gramática pueda incluir esa sentencia en su lenguaje.

8. **[10 pt.]** Explique si el siguiente código da un error de compilación y por qué sí o por qué no.

```
class trabajador
{
  public:
     void hora_de_levantarse( )
     { .... ... .... }
};
class estudiante
{
    void hora_de_levantarse( )
     { .... ... .... }
};
class ayudante_alumno : public trabajador, public estudiante
{

};

int main()
{
    ayudante_alumno obj;
}
```

9. **[10 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)
    {
    }
};
```

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
```