Paradigmas de la Programación – Primer Parcial

23 de Abril de 2015

Apellido y Nombre: _____

1. **[10 pt.]** Seleccione todas las respuestas correctas entre las diferentes opciones que se dan para completar cada oración:

   a) Una gramática independiente de contexto (*context-free grammar*)...
      1) tiene predicados.
      2) es inambigua.
      3) tiene reglas de reescritura.
      4) describe la sintaxis de un lenguaje.
      5) es más expresiva que una gramática dependiente del contexto.
      6) es tan expresiva como una máquina de Turing.

   b) Las variables están compuestas de... (elija la opción más precisa)
      1) un nombre y un valor.
      2) un l-valor y un r-valor.
      3) un identificador y un valor.
      4) un identificador que está ligado a un valor que está ligado a una dirección en memoria.
      5) un identificador que está ligado a una dirección en memoria que contiene un valor.
      6) una dirección de memoria, un valor y un identificador en el texto del programa.

   c) La clausura de una función es necesaria para...
      1) definir funciones en un alcance global
      2) mantener la referencia a las variables locales de la función si tenemos alcance dinámico
      3) mantener la referencia a las variables locales de la función si tenemos alcance estático
      4) mantener la referencia a las variables libres de la función si tenemos alcance dinámico
      5) mantener la referencia a las variables libres de la función si tenemos alcance estático

   d) La diferencia entre polimorfismo y sobrecarga es...
      1) que sobrecarga se aplica sólo a algunos tipos, mientras que polimorfismo es más general.
      2) que la sobrecarga se comprueba en tiempo de ejecución y el polimorfismo en tiempo de compilación.
      3) que en la sobrecarga tenemos diferentes implementaciones para un mismo símbolo y en el polimorfismo tenemos una sola implementación con tipos generales.
      4) que la sobrecarga usa tipos concretos mientras que el polimorfismo usa variables de tipo.
      5) que la sobrecarga la analiza el compilador y el polimorfismo no.

   e) Las excepciones se pueden usar para optimizar código...
      1) porque tienen alcance dinámico.
      2) porque son saltos explícitos entre partes del programa.
      3) porque son imperativas.
      4) porque son funcionales.

2. **[20 pt.]** Muestre con un ejemplo que la siguiente gramática es ambigua, y modifíquela para que se convierta en una gramática inambigua.

```
<a> ::= <a> <b> | <b>
<b> ::= <hoja> | <c> | <a>
<c> ::= bin <d> <b> | bin <d> <b> ban <b>
<hoja> ::= 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g'
```

3. **[30 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.

   Nota de estilo: El nivel de granularidad de la descripción puede limitarse a bloques del texto del programa, no es necesario diagramar como estados distintos de la ejecución las instrucciones que no apilen a otro bloque. Si lo desea, puede usar índices en el mismo texto del programa para una representación más compacta de la pila.

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

4. **[20 pt.]** En el siguiente programa tipo Algol,

```
begin
  integer n;
  procedure p(k: integer);
    begin
      k := k+3;
      print(n);
      n := n+(2*k);
    end;
  n := 0;
  p(n);
  print(n);
end;
```

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

   a) por valor?

   b) por valor-resultado?

   c) por referencia?

5. **[20 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*y;
```