Paradigmas de Programación: Parcial 2

4 de junio de 2007

1. Defina como parte del lenguaje de kernel las operaciones $X = @C$, $C := X$, $X = C := Y$ y de la semántica para cada una de ellas en la máquina abstracta.

2. Defina un tipo de datos FIFO que se comporte como una cola First In First Out, que defina las siguientes operaciones: *NewFIFO getElem*, *putElem* y *isEmpty*. La definición del tipo debe ser abierta, con estado y empaquetada.

3. Defina un mecanismo de pasaje de argumentos donde el argumento es una funcion que contiene al identificador de celda (caso similar al de pasaje por ~~valor resultado~~ nombre) pero, además se requiere que la función se ejecute si y solamente si, el cuerpo del procedimiento necesita del nombre de la celda. Utilize las primitivas de disparadores disponibles en nuestro lenguaje de kernel para resolver este problema.

4. En el teorico, las nuevas instancias de objetos son procedimientos que responden a mensajes. Redefina este concepto para que las nuevas instancias sean records donde para acceder a sus procedimientos se utilice la deferencia por punto. Ayuda: Compare las implementaciones de pila de la página 424 para el caso "seguro, con estado y empaquetadas *sin* despachador de procedimiento" con "seguro, con estado y empaquetadas *con* despachador de procedimiento"

5. En el práctico 7 y en el capítulo 3 se muestran distintas formas de recorrer un árbol. Una forma alternativa:

```
fun {InOrder T}
   case T of tree(V L R) then
      {Append {InOrder L} V|{InOrder R}}
   [] leaf then
      nil
   end
end
```

   Escribir una versión iterativa de la función `InOrder`. Dar la invariante de corrección.

6. Dado el siguiente bloque de código:

```
fun lazy {MakeW} {Delay 1000} 1 end
fun lazy {MakeX} {Delay 2000} 2 end
fun lazy {MakeY} {Delay 3000} 3 end
fun lazy {MakeZ} {Delay 4000} 4 end
W={MakeW}
X={MakeX}
Y={MakeY}
Z={MakeZ}
R = W+X+Y+Z
```

   a) Reordenar los elementos de la suma asignada a `R` (pueden también usarse paréntesis para asociar) de forma tal que se minimice el tiempo de ejecución. Indicar cómo queda la asignación, cuanto tiempo de ejecución resulta, y por qué es el mínimo posible.

   b) Reordenar los elementos de la suma asignada a `R` (pueden también usarse paréntesis para asociar), y agregar bloques `thread ... end` dentro de la instrucción, de forma tal que se minimice el tiempo de ejecución. Indicar cómo queda la asignación, cuanto tiempo de ejecución resulta, y por qué es el mínimo posible.

7. Definir una función `Curry` que tome como argumento una función de dos argumentos de Oz, y devuelva su versión curryficada (una función de un argumento que devuelve una función de un argumento). Es decir, `{F X Y}` debería ser equivalente a `{{{Curry F} X} Y}` para cualesquiera `F`, `X` e `Y`.