Paradigmas de Programación
Parcial 1

18 de Abril de 2006

Ejercicios

0% 1. Extendiendo el manejo de Excepciones: Una sentencia **try** se puede especificar una clausula **finally** que siempre será ejecutada, independientemente de que una excepcion sea levantada por la sentencia del **try**. La nueva sintaxis

$$\textbf{try } \langle S \rangle_1 \textbf{ finally } \langle S \rangle_2 \textbf{ end}$$

se traduce al lenguaje de kernel como:

try $\langle S \rangle_1$
catch X **then**
  $\langle S \rangle_2$
  raise X **end**
end
$\langle S \rangle_2$

El identificador X se elije de manera que de que *no* sea libre en $\langle S \rangle_2$.

a) ¿Por que X debe ser una variable *no* libre en $\langle S \rangle_2$?

b) Defina una traducción en la que $\langle S \rangle_2$ aparezca solo una vez.

80% 2. La sintáxis concreta que se dio para el lenguaje de kernel es una de muchas posibles. Se podría hacer un lenguaje muy similar, con las mismas primitivas y semántica, pero cuya notación sea distinta. Tomemos la siguiente instrucción en lenguaje de kernel:

```
Max = proc {$ X Y ?Z}
   local M in
      {>= X Y M}   % Recordar que esto es M = (X>=Y)
      if M then
         Z = X
      else
         Z = Y
      end
   end
end
```

Supongamos ahora que definimos el "nuevo lenguaje de kernel" que tenga los siguientes cambios en sus definiciones sintácticas:

```
<procedure> ::= '(' 'proc' '('<x1> ... <xn> ')' <s> ')'
<s> ::= '(' 'local' <x> <s> ')'
      | '(' 'if' <x> '(' <s1> ')' '(' <s2> ')' ')'
      | '(' 'set' <x1> <x2> ')'
      | ...
```

Es decir, cambiamos la notación para procedimientos, definición de variables locales, condicional, y binding de variable a variable (los puntos suspensivos denotan las otras instrucciónes, que dejamos como en el lenguaje kernel del libro.

Traducir la instrucción que se dió al principio de este punto para que esté escrita en el "nuevo lenguaje de kernel".

0% 3. Leer con atención el siguiente programa:

```
local P Q in
   proc {Q} {Browse hola} end
   proc {P Q} {Q} end
   local Q in
      proc {Q} {Browse chau} end
      {P Q}
   end
end
```

a) ¿Que resultado se muestra en el browser, en lenguaje kernel (con scope estático)?

b) ¿Que resultado se mostraría en el browser, si el lenguaje kernel tuviera scope dinámico?

Justificar en ambos casos la respuesta describiendo el estado del entorno justo antes de la llamada a P, durante la ejecución de P, y explicando porque el entorno tiene ese estado.

20% 4. Dado el programa

```
local P X in
   proc {P A B ?R}
      if B==0 then
 R=A
      else
 {P A+1 B-1 R}
      end
   end
   {P 1 2 X}
end
```

a) Indicar paso a paso el estado del stack cuando se ejecuta. Notar que puede ser necesaria una traducción a lenguaje de kernel.

b) ¿Que valor tiene X al terminar de ejecutarse la llamada a procedimiento?