Universidad de Las Palmas de Gran Canaria
Escuela de Ingeniería Informática
Examen de Programación IV de Programación Funcional y Lógica
18 de mayo de 2015

Nombre y apellidos: .....

Notas:

- La duración del examen es de 50 minutos.
- Permanezca en su lugar. Para hablar con el profesor, pida permiso para levantarse.
- Debe tener visible sobre la mesa un documento que le identifique.
- No se permite el acceso a otra información que no sea este enunciado.
- El examen se realizará utilizando únicamente la página de edición correspondiente en el servidor aulaga.
- Sólo se permite un lápiz o bolígrafo durante la realización el examen.
- Si necesita escribir anotaciones, use el reverso de esta hoja, que deberá entregar al finalizar.
- Antes de comenzar el examen escriba su nombre arriba.

1) (2.5 puntos) Defina en Scheme una función de nombre **`final`** a la que se le pasa una lista y devuelve una lista con los elementos iguales al final de la lista (vacía para una lista vacía). Se pueden crear y emplear funciones auxiliares.
   Ejemplos:
   `(final '(a c a d a e f b))` *devuelve:* **`'(b)`**
   `(final '(a c c d a e f b b b))` *devuelve:* **`'(b b b)`**

2) (2.5 puntos) Defina una función de nombre **`profundidad`** a la cual se le pasa como parámetro un átomo o una lista que puede contener átomos (no lista) y listas recursivamente (ver ejemplos). La función devuelve la profundidad máxima del dato pasado, teniendo en cuenta que un átomo (no lista) tiene profundidad 0 y la profundidad de una lista se mide como 1 más la profundidad de su elemento con más profundidad. Se recuerda que existe la función **list?.**
   Ejemplos:
   `(profundidad 'a)` *devuelve:* **`0`**
   `(profundidad '())` *devuelve:* **`1`**
   `(profundidad '(c a b (r t) f (1) k))` *devuelve:* **`2`**
   `(profundidad '(c a b (r t) f (1 (3 4 5)) t))` *devuelve:* **`3`**
   `(profundidad '(c a b (r t) f (1 ((3 4) 5)) 4))` *devuelve:* **`4`**

3) (2.5 puntos) Defina en Prolog un predicado **`final(`***L*`, `*P*`)`** tal que, dada una lista *L*, calcule una lista *P* con los elementos iguales al final de la lista *L*. Si la lista está vacía *P* será la lista vacía. Se pueden crear y emplear predicados auxiliares.
   Ejemplo:
   `?- final([a, c, a, d, a, e, f, b, b, b], N).`
   responde:
   `N = [b, b, b]`

4) (2.5 puntos) Defina un predicado **`profundidad`**(*L*, *N*) tal que, dado un dato que puede ser un átomo (no lista) o una lista que puede contener átomos y listas recursivamente (ver ejemplos), *N* toma el valor de la profundidad máxima en *L*, siendo 0 para un átomo (no lista) y para una lista 1 más la profundidad de su elemento con más profundidad. Se puede usar el predicado **`is_list`**(*L*) que es verdadero si *L* es una lista. Se puede usar la función **`max`**(*N1*,*N2*), está función sólo es usable en la parte derecha de "**`is`**" en una evaluación aritmética. También es posible crear un predicado propio **`máximo`**.
   Ejemplo:
   `?- profundidad(c, N).` *responde:* **`N = 0`**
   `?- profundidad([], N).` *responde:* **`N = 1`**
   `?- profundidad([c, a, b, a], N).` *responde:* **`N = 1`**
   `?- profundidad([c, a, b, [r, t], f, k], N).` *responde:* **`N = 2`**
   `?- profundidad([c, a, b, [r, t], f, [1, [[3, 4], 5]], q], N).` *responde:* **`N = 4`**