Examen conv. 3 curso 96-97 8 Sept 1997 1 hora 15 minutos

Apellidos y Nombre:
.....

Problema 1: Representación basada en frames/Common Lisp/Prácticas

[35 puntos]

El siguiente código corresponde a la base de datos sobre apartamentos del programa que has utilizado en la práctica 4.

```
    (form :is-a nil :name 'object)
    (form :name 'cosa
          :is-a 'object)
    (form :name 'vivienda
          :is-a 'cosa
          :slots (list (list 'ruido-calle 'if-needed #'calcula-ruido)
                       (list 'propietario)
                       (list 'calle-nombre)
                       (list 'calle-numero)
                       (list 'piso)
                       (list 'color-pared)
                       (list 'material-suelo)
                       (list 'numero-habitaciones)
                       (list 'distancia-carretera)))
    (form :name 'apartamento
          :is-a 'vivienda
          :slots '((numero-habitaciones = 2)
                   (color-pared = blanco)
                   (material-suelo = gres)
                   (piso)
                   (mano)))
    (form :name 'chalet
          :is-a 'vivienda
          :slots (list (list 'ruido-calle
                             'if-needed #'calcula-ruido-en-casa)
                       (list 'num-pisos '= '2)))
    (form :name 'apt-breton-22
          :is-a 'apartamento
          :slots '((calle-nombre = Breton)
                   (calle-numero = 22)
                   (material-suelo = madera)))
    (form :name 'apt-22-1
          :is-a 'apt-breton-22
          :slots '((numero-habitaciones = 3)
                   (mano = derecha)
                   (piso = 2)))
    (form :name 'apt-22-2
          :is-a 'apt-breton-22
          :slots '((numero-habitaciones = 4)
                   (mano = izquierda)
                   (piso = 2)))
))
...

(defun calcula-ruido (form slot)
  (let* ((valor (find-aspect-from-supers form 'ruido-calle '=)) piso)
    (cond (valor valor)
          (t (setq piso (get-value form 'piso))
             (cond ((> piso 15) 'muy-bajo) ((> piso 8) 'bajo)
                   ((> piso 4) 'medio) ((> piso 1) 'alto)
                   (t 'muy-alto))))))
```

NOTA - Al hacer los siguientes apartados pon especial cuidado en que el código que escribas refleje tu conocimiento sobre frames, sea el m'nimo y resulte lo mas "elegante" posible.

(a - 15 puntos) Escribe el código necesario para que siempre que se pida el valor del slot `propietario` de una vivienda (`(get-value 'apt-22-1 'propietario)`) ocurra lo siguiente>

- se incremente un contador que lleve la contabilidad del número de veces que se ha pedido dicho valor de dicha vivienda
- se escriba dicho valor en el dispositivo de salida

p.e.

```
(get-value 'apt-22-1 'propietario)
dispositivo de salida--> 1
(get-value 'apt-22-1 'propietario)
dispositivo de salida--> 2
(get-value 'apt-22-2 'propietario)
dispositivo de salida--> 1
```

(b - 20 puntos) Haz las modificaciones de código necesarias para propagar la contabilidad de acceso al slot `propietario` a todos sus antecesores en la jerarquía. De esta forma, al acceder a cualquier objeto descendiente de `object` se escribir'a en el dispositivo de salida la suma de accesos correspondientes a todos sus descendientes.
p.e.

```
(get-value 'apt-22-2 'propietario)
```

dispositivo de salida--> 2

```
 (get-value 'vivienda 'propietario)
```

dispositivo de salida--> 5

Problema 2: Búsquedas informadas *[40puntos]*

Considérese el siguiente problema con fichas deslizantes. Hay 3 fichas negras (N), 3 fichas blancas (B) y una casilla vacía (V) en la siguiente posición inicial:

N N N B B B V

Los movimientos permitidos son los siguientes:

1. Una ficha puede moverse a una celda adyacente vacía, con coste igual a la unidad.
2. Una ficha puede desplazarse a una celda vacía saltando dos fichas como máximo, con un coste igual al número de fichas que haya saltado.

El objetivo del puzzle consiste en colocar todas las fichas blancas a la izquierda de las fichas negras, independientemente de la posición de la celda vacía.

Se pide especificar una función heurística h para este problema y mostrar el árbol de búsqueda producido por el algoritmo A* utilizando dicha función heurística. (Deja claro los operadores utilizados. Para cada nodo debe quedar claramente indicado el operador utilizado para su generación, los distintos valores utilizados para evaluarlo y el orden de evaluación. Parar el dibujo del árbol con el primer nodo que alcance profundidad **5** sin contar el inicial.)

Problema 3: Sistemas basados en reglas/CLIPS *[25 puntos]*

Completa el siguiente programa CLIPS. El programa pregunta por un color de ojos y visualiza todas las personas con el color elegido. A continuación pregunta por una edad y visualiza las personas con el color de ojos y la edad elegidad anteriormente.

```
(deftemplate persona (slot nombre)
                     (slot ojos)
                     (slot edad))

(deftemplate tarea (slot tipo (allowed-symbols inicia busca-ojos busca-edad))
                   (slot color (allowed-symbols azul verde negro))
                   (slot edad (type INTEGER)))

(deffacts inicio "define el estado inicial"
        (tarea (tipo inicia))
        (persona (nombre Jose)  (ojos verde) (edad 18))
        (persona (nombre Antonio)  (ojos azul) (edad 32))
        (persona (nombre Ramon)  (ojos azul) (edad 27)))

(defrule inicializa
        ?h1 <- (tarea (tipo inicia))
=>
        (printout t "color de los ojos?:")
        (bind ?color (read))
        (assert (tarea (tipo busca-ojos) (color ?color)))
        (printout t "* Personas con color de ojos " ?color ": " crlf)
        (retract ?h1))

(defrule busca-color-ojos   "muestra las personas con el color de ojos elegido"
        ?h1 <- (tarea (tipo busca-ojos) (color ?color))
        ……)
=>
        (printout t ?nombre " tiene los ojos " ?color crlf))

(defrule pasa-a-ultima-tarea "cambia de tarea"
        ?h1 <- (tarea (tipo busca-ojos))
=>
        (printout t "* De los anteriores visualizo los mayores de?:")
        (bind ?edad …….)
        (modify ……)

(defrule muestra-color-edad "muestra personas con el color de ojos y
                             mayores que la edad elegida"
        ?h1 <- (tarea (tipo busca-edad) (color ?color) (edad ?edad))
        (persona (nombre ?nombre) (ojos ?color) (edad ?edad-persona))
        (test ……)
=>
        (printout t ?nombre " tiene los ojos " ?color " y tiene mas de "
                    ?edad crlf))
```

NOTAS:

- `(read),` devuelve el símbolo o el entero leido del teclado.
- La estrategia de resolucion de conflictos elegirá primero las reglas más especificas, es decir, aquellas que tengan mayor numero de precondiciones. (NO hace falta dar prioridad a las reglas).

Ejemplo de ejecucion:

```
CLIPS> (reset)
CLIPS> (run)
color de los ojos?:azul
* Personas con color de ojos azul:
Ramon tiene los ojos azul
Antonio tiene los ojos azul
* De los anteriores visualizo los mayores de?:28
Antonio tiene los ojos azul y tiene mas de 28
```