Lenguajes de Inteligencia Artificial

Segundo curso de
Ingeniería Técnica en Informática de Gestión e
Ingeniería Técnica en Informática de Sistemas
Escuela Politécnica Superior
Universidad de Córdoba

Examen correspondiente a la convocatoria de febrero: 8 de febrero de 2005

SCHEME

1. Codifica un predicado denominado ***bisiesto?*** que reciba como parámetro a un ***número*** y determine si corresponde o no a un año bisiesto, teniendo en cuenta que:
   - Un año es bisiesto si es divisible por 4 pero no es divisible por 100:
     (bisiesto? 2008) => #t
   - Un año es bisiesto si es divisible por 100 y además es divisible por 400. Por ejemplo: *(bisiesto? 1600) => #t*

   Nota: por tanto, los años que son divisibles por 100 pero no son divisibles por 400 no son bisiestos. Por ejemplo: *(bisiesto? 1900) => #f*

   1 punto

2. Codifica una función iterativa denomina ***suma-diferencia-absoluta*** que reciba como parámetros a dos funciones ***f*** y ***g*** y tres números denominados ***inferior***, ***superior*** y ***paso*** y que permita calcular la suma de la siguiente serie numérica:

$$\sum_{\substack{x=inferior \\ x \leftarrow x+paso}}^{superior} \frac{|f(x)-g(x)|}{2}$$

   1'5 puntos

3. Codifica una función recursiva denominada **agrupar** que reciba como parámetro a una ***lista*** con elementos y devuelva otra lista compuesta por sublistas, donde cada una de ellas ha de contener a un elemento de la lista original y el número de veces que aparece en ella. Por ejemplo.
   (agrupar '(a b c a d b a a d)) => ((a 4) (b 2) (c 1) (d 2))
   Nota: se recomienda usar funciones auxiliares para ***contar*** el número de veces que aparece un elemento en una lista y ***suprimir*** un elemento de una lista:
   (contar 'a '(a b c a d b a a d)) => 4
   (suprimir 'a '(a b c a d b a a d)) => (b c d b d)

   2'5 puntos

4. El tipo abstracto de datos ***pueblo*** consta de los campos ***nombre***, ***provincia*** y ***habitantes***.
   - Utiliza las listas de asociación para codificar "todas" las funciones de creación, consulta y modificación de este tipo abstracto de datos.
   - Utiliza las funciones anteriores para codificar otra función denominada ***escribir-pueblos-provincia*** que reciba como parámetros una ***lista*** de pueblos, el nombre de una ***provincia*** y el nombre de un fichero y que escriba en el fichero el nombre y los habitantes de cada uno de los pueblos de dicha provincia.

   Por ejemplo:

```
(escribir-pueblos-provincia
        '(
        ((nombre "Luque") (provincia "Córdoba") (habitantes 8000))
        ((nombre "Loja") (provincia "Granada") (habitantes 20000))
        ((nombre "Añora") (provincia "Granada") (habitantes 3000))
        ((nombre "Rota") (provincia "Cádiz") (habitantes 30000))
        )
        "Córdoba" "salida.txt")
```

   El fichero ***salida.txt*** debe contener la siguiente información

```
Luque 8000
Añora 3000
```

   2 puntos

PROLOG

5. Predicados relativos a pueblos y provincias:
   - Utiliza el predicado ***situada_en(Pueblo,Provincia)*** para declarar los siguientes hechos.
     - *Baena, Zuheros y Luque están en la provincia de Córdoba*
     - *Loja e Iznalloz están en el provincia de Granada*
   - Codifica un predicado denominado ***total_pueblos_provincia*** que posea dos parámetros: el primero será el nombre de una ***provincia*** y el segundo será el ***número*** de pueblos que están en esa provincia.
     ?-total_pueblos_pronvicia("Granada",N).
     N = 2

   Nota: Utiliza el predicado ***bagof*** y un predicado auxiliar para ***contar*** los elementos de una lista.

   1 punto

6. Codifica en Prolog el ejercicio número 1 de Scheme, es decir, codifica un predicado denominado ***bisiesto*** que determine si un año es bisiesto o no. Por ejemplo:
   ?- bisiesto(2008).
   Yes
   ?- bisiesto(1600).
   Yes
   ?- bisiesto(1900).
   No

   1 punto

7. Codifica en Prolog el ejercicio número 3 de Scheme, es decir, codifica un predicado denominado ***agrupar*** que reciba dos parámetros, el primero será un lista con elementos y el segundo será la lista con los elementos agrupados. Por ejemplo:
   ?- agrupar([a,b,c,a,d,b,a,a,d], R).
   R = [[a,4],[b,2],[c,1],[d,2]]
   Nota: se recomienda usar predicados auxiliares para ***contar*** el número de veces que aparece un elemento en una lista y ***suprimir*** un elemento de una lista:
   ?- contar(a, [a ,b, c, a, d, b, a, a, d],N) .
   N = 4
   ?- suprimir(a,[a, b, c, a, d, b, a, a, d],R).
   R = [b, c, d, b, d]

   1 punto