



Programación Declarativa

Examen final - Teoría

NOMBRE:

APELLIDOS:

GRUPO:

NORMAS DE REALIZACIÓN:

- Escribir el nombre, apellidos y grupo al que pertenece en los campos anteriores.
- Señale la respuesta correcta en esta página.
- El tiempo de realización de este test es de 20 minutos.
- La valoración del test en el conjunto del examen es de 2 puntos (sobre 10).

EJERCICIO 1

Para el objetivo $X=1+2$, $Y:=X$, Prolog da como resultado

- (a) Yes.
- (b) No.
- (c) Da un error.
- (d) $X=1+2$, $Y = 1+2$.

EJERCICIO 2

Considere los siguientes objetivos en Prolog

- (1) $X \setminus= 3$, $X=5$, X is $3+2$.
- (2) $X=3+2$, $Y=5$, $X=Y$.
- (3) $X=5$, $X= Y$, $Y=5$.
- (4) $X=5$, $Y=5$, $X= Y$.

¿Cuál de las siguientes afirmaciones es correcta ?

- (a) La solución de (1) es Yes y las soluciones de (2) y (3) son No.
- (b) La solución de (3) y (4) es $X=5$, $Y=5$.
- (c) La solución de (2) es No.
- (d) La solución de (2) es Yes.

EJERCICIO 3

En Prolog, al ejecutar `[1 | [2 | []]] = [X, Y]` se obtiene:

- (a) X=1, Y=[2].
- (b) X=1, Y=2.
- (c) No.
- (d) error (expresión incorrecta)

EJERCICIO 4

El predicado **descarta(E1,L1,L2)**, descrito a continuación, debería ser verdadero para las ternas de argumentos tales que **L2** es la lista resultante de eliminar todas las ocurrencias del elemento **E1** de la lista **L1**:

- `descarta(_, [],[])`.
- `descarta(E1, [E1|R], Res):- descartar(E1, R, Res)`.
- `descarta(E1, [C|R], [C|R2]):- descartar(E1, R, R2)`.

Sin embargo, no es correcto ya que es verdadero para valores de **L2** que se corresponden con el resultado de eliminar de **L1** todas, algunas o ninguna de las ocurrencias de **E1**. Para resolver el problema que se plantea

- (a) bastaría con añadir un corte a la línea 1.
- (b) bastaría con añadir un corte a la línea 2.
- (c) bastaría con añadir un corte a la línea 3.
- (d) no bastaría con añadir cortes.

EJERCICIO 5

En Haskell, según la estrategia de curriificación, dada una función con la siguiente declaración de tipos **f:: Int->Int->Int**, el resultado de aplicar esta función a un argumento será

- (a) una función de tipo `Int->Int`
- (b) una función de tipo `Int->Int->Int`
- (c) una función con dos argumentos
- (d) ninguna de las afirmaciones anteriores es correcta

EJERCICIO 6

En Haskell: `tail [1,2,'s',[a,s]]` es igual a

- (a) `[2,'s',[a,s]]`
- (b) `(2,'s',[a,s])`
- (c) `1: [2,'s',[a,s]]`
- (d) ninguna de las anteriores

Programación Declarativa

Examen final

NORMAS DE REALIZACIÓN:

- Cada ejercicio deberá contestarse en hojas diferentes
- Cada hoja deberá contener el nombre y apellidos del alumno, así como el grupo al que pertenece.
- El tiempo de realización es de 2 horas y 30 minutos

EJERCICIO 1 (1.5 puntos)

El método *checksum*, utilizado para control de errores en comunicaciones, consiste en añadir un último dígito a la cadena binaria a enviar. Este dígito adicional se calcula como la suma binaria (sin acarreo) de todos los dígitos de la cadena. La definición de la suma binaria es la siguiente:

- $\text{sumabinaria}(0,0,0)$.
- $\text{sumabinaria}(0,1,1)$.
- $\text{sumabinaria}(1,0,1)$.
- $\text{sumabinaria}(1,1,0)$.

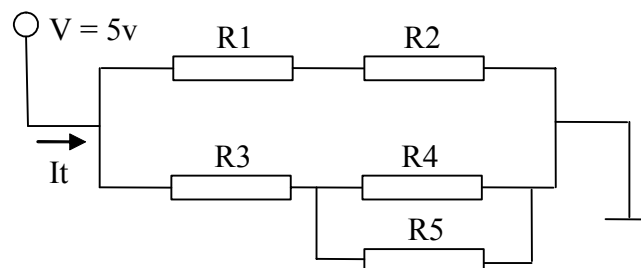
Realizar el predicado **valido(Cadena)**, donde **Cadena** es una lista de 0s y 1s, que verifique que el último elemento de la lista es igual al checksum de todos los anteriores.

EJERCICIO 2 (2 puntos)

La siguiente figura muestra un esquema de representación en forma de árbol (basado en los predicados **serie/2** y **paralelo/2**) que permite definir circuitos eléctricos formados por resistencias:



Considere el siguiente circuito, donde los valores de las resistencias son los siguientes: $R_1=1000 \Omega$, $R_2=1000 \Omega$, $R_3=1000 \Omega$, $R_4=2000 \Omega$, $R_5=1000 \Omega$.



Cuestiones:

- (a) Representélo en Prolog.
- (b) Describa en Prolog un predicado que calcule la resistencia total del circuito.

NOTA: Resistencia en serie $R_{\text{serie}} = R1 + R2$

Resistencia en paralelo $R_{\text{paralelo}} = 1 / (1/R1 + 1/R2)$

EJERCICIO 3 (1.5 puntos)

Considerando la siguiente base de conocimientos:

- $p(X,Y):-q(X),r(Y)$.
- $q(a):-r(2)$.
- $q(b):-r(1),!,fail$.
- $q(_):-!$.
- $r(1)$.
- $r(2):-!$.
- $r(3)$.

Dibuje el árbol de deducción para el objetivo $p(X,Y)$.

EJERCICIO 4 (2 puntos)

El mundo de los bloques consiste en un plano sobre el cual se encuentran una serie de bloques. Este mundo se caracteriza por:

- Un bloque puede estar en el suelo o bien apilado sobre otro bloque.
- Un bloque que no tiene otro/s bloque/s apilados sobre él se considera libre.
- Un bloque se puede apilar sobre otro o bien desapilar:
 - Apilar A sobre B:
 - Precondiciones: A y B son diferentes y los bloques A y B están libres.
 - Poscondiciones: El bloque A queda apilado sobre el bloque B.
 - Desapilar A:
 - Precondiciones: El bloque A está libre y no está en el suelo.
 - Poscondiciones: El bloque A queda en el suelo.

El objetivo que perseguimos es pasar de una configuración del mundo de bloques a otra utilizando las operaciones permitidas sobre los bloques.

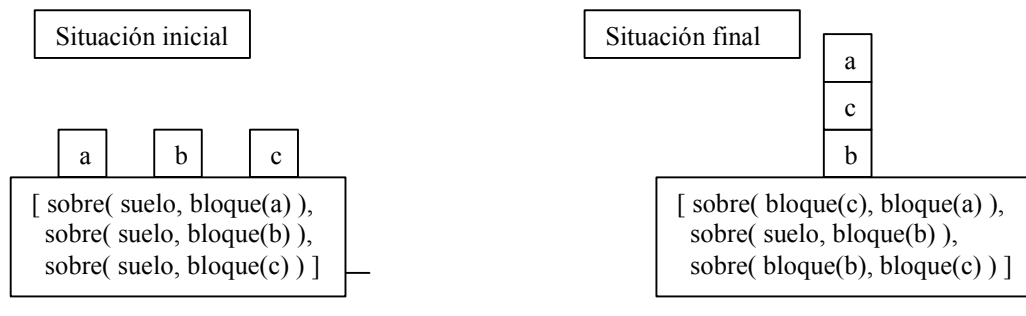
Representación:

Para cada bloque tendremos un término **bloque/1** en nuestra base de conocimiento.

Para representar un estado se utilizará una lista de términos **sobre/2**.

Para cada bloque habrá en la lista un término **sobre/2** de modo que el segundo argumento nos marca el bloque al que nos referimos y el primero será un término que nos marca sobre qué se encuentra el bloque en cuestión. El primer término podrá ser bien suelo o bien un término de **bloque/1**.

Consideraremos el siguiente problema:



Cuestiones:

(a) Programe en Prolog los hechos de la base de conocimiento necesarios para la resolución de este problema, considerando los predicados **bloque**, **inicial** y **final**.

(b) Programe en Prolog el predicado **libre** y los predicados de los operadores.

Nota: los siguientes predicados están ya programados:

- **esta(Elemento, Lista)** : verdadero si la lista **Lista** contiene el elemento **Elemento**.
- **insertar(Elemento, Lista1, Lista2)** : verdadero para las ternas de argumentos tales que **Lista2** es una lista que contiene los elementos de **Lista1** junto con el elemento **Elemento**.
- **eliminar(Elemento, Lista1, Lista2)** : verdadero para las ternas de argumentos tales que **Lista2** es una lista que contiene los elementos de **Lista1** habiendo eliminado la primera ocurrencia del elemento **Elemento**.

EJERCICIO 5 (1 punto)

Definición de funciones en Haskell.

(a) Definir en Haskell la función **multiplos** que utilice como entradas un número entero **a** y una lista de números enteros **x** y genere como salida una lista de números enteros formada por los múltiplos de **a** que se encuentran en **x**. (Nota: la función **mod a b** devuelve el resto de la división entera entre **a** y **b**).

(b) Definir en Haskell la función **potencia** que utilice como entradas dos números enteros, **a** y **b**, y genere como salida **a** elevado a **b**, utilizando tan sólo el producto y la recursividad.