Guía de Trabajos Prácticos VIII – Programación Lógica IV

1) Dado el siguiente programa en Prolog que calcula el factorial de un número:

```
factorial(0, 1).
factorial(Numero, Factorial) :-
        Numero > 0,
        NumeroAnt is Numero - 1,
        factorial(NumeroAnt, FactorialAnt),
        Factorial is Numero * FactorialAnt.
```

El resultado de evaluar "factorial(5, X)." es:

```
X = 120;
false.
```

Utilice un operador de corte donde corresponda para que el programa finalice una vez terminada la recursión y no retorne el "false" final.

2) Dado el siguiente programa en Prolog:

```
p(1).
p(2):-!.
p(3).
```

Evaluar las siguientes consultas, dejando registrado para cada una el resultado obtenido, y en lenguaje natural, por que se obtiene dicho resultado detallando los puntos de elección que han sido desechados y cual es el operador de corte por el cual han sido quitados.

a) p(X).
b) p(X), p(Y).
c) p(X), !, p(Y).

3) Implementar un predicado "eliminar_primero/3" que quite de una lista la primera aparición de un determinado elemento, utilizar el operador de corte donde considere necesario para garantizar que no se realicen unificaciones innecesarias.

Ej:
eliminar_primero([1, 2, 3, 1, 2, 3], 2, X).
X = [1, 3, 1, 2, 3].

eliminar_primero([1, 2, 3, 1, 2, 3], a, X).
X = [1, 2, 3, 1, 2, 3].

4) Implementar un predicado "agregar_nuevo/3" que agregue a una lista un elemento, solo si la misma no lo contiene. Utilizar corte para no proceder en caso de detectar que el elemento ya existe.

Ej:
agregar_nuevo(a, [1, 2, 3], X).
X = [1, 2, 3, a].

agregar_nuevo(1, [1, 2, 3], X).
X = [1, 2, 3].

5) Dadas dos palabras, representadas como listas de caracteres, evaluar la semejanza entre las mismas. Para esto se verificarán, por posición, las letras de las dos palabras, cada coincidencia sumará un punto y cada vez que las letras no coincidan se restará un punto.

ejemplo:
semejanza([h,o,l,a], [h,o,l,o], S). --> S = 2.
semejanza([m,e,s,a], [m,e,s,a,d,a], S). --> S = 2.
semejanza([s,o,l,a], [m,o,n,a], S). --> S = 0.
semejanza([s,o,l], [c,a,s,a], S). --> S = -4.

6) Dado un diccionario en forma de lista de palabras, se quiere buscar una palabra cualquiera, para lo cual se deberá chequear que la misma pertenezca al diccionario. Si existe se debe devolver una lista de un elemento con la palabra buscada, si no existe, se debe devolver una lista con las alternativas a la palabra buscada y su valoración de semejanza.

Las alternativas devueltas en caso de no existir la palabra en el diccionario, serán tomadas del mismo en virtud de la semejanza con la palabra buscada. Se tomarán como semejantes aquellas palabras cuyo valor de semejanza sea mayor a cero.

Para el desarrollo de ésta segunda parte del programa, se puede usar el predicado predefinido "atom_chars(C, L)" que toma un átomo (C) y lo transforma en una lista de literales (L): atom_chars(hola, L). --> L = [h, o, l, a].

La definición del diccionario en principio es:
dic([sanar, hola, sabana, sabalo, prueba, computadora, cartera, mate, termo, mesa, silla, sarna]).

buscar(hola, L). --> L = [hola]
buscar(holo, L). --> L = [[hola, 2]]
buscar(saban, L). --> L = [[sanar, 1], [sabana, 4], [sabalo, 2]]

7) Escriba un predicado en prolog de aridad 6 que implemente la función reemplazar, la cual recibirá los siguientes parámetros:

1- el elemento a reemplazar en la primer lista
2- el elemento de reemplazo
3- a partir de que instancia encontrada comienza a reemplazar, debe validarse que su valor sea >= 1
4- cuantos reemplazos como máximo se harán, siendo -1 el valor para indicar que reemplace todas las instancias que encuentre. Debe validarse que su valor sea X= -1 o X >=1
5- una lista de elementos en donde se reemplazará
6- el último parámetro debe unificar con la lista resultante

Nota: Se debe implementar la funcionalidad, no se puede usar un predicado predefinido que la implemente.
Ej:
reemplazar(1, a, 0, -1, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
El valor de inicio debe ser mayor o igual a 1
Fail.

reemplazar(1, a, 1, -2, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
La cantidad de reemplazos debe ser -1 o mayor o igual 1
Fail.

reemplazar(1, a, 1, -1, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
L = [a, a, 2, 2, 3, 3, 2, 2, a, a]

reemplazar(1, a, 1, 1, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
L = [a, 1, 2, 2, 3, 3, 2, 2, 1, 1]

reemplazar(1, a, 2, 2, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
L = [1, a, 2, 2, 3, 3, 2, 2, a, 1]

reemplazar(1, a, 2, -1, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
L = [1, a, 2, 2, 3, 3, 2, 2, a, a]

reemplazar(1, a, 5, -1, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
L = [1, 1, 2, 2, 3, 3, 2, 2, 1, 1]

reemplazar(4, a, 1, -1, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
L = [1, 1, 2, 2, 3, 3, 2, 2, 1, 1]

6) Cree un programa que encuentre la salida dado el plano de una casa, para lo cual:

a) Debe resolver como modelar el plano.
b) Debe encontrar la forma de avanzar en el camino hacia la salida.
c) Debe verificar que no se formen bucles en el camino.

Ej.:

salir(a, Camino).
Camino = [a, b, c, g, Salida];
Camino = [a, b, e, f, g, Salida];
Camino = [a, d, f, g, Salida];
Camino = [a, d, f, e, b, c, g, Salida].