PARTE 1: Encuestas

Se dispone de dos encuestas sobre los equipos y jugadores de fútbol más populares. Los equipos están almacenados en hechos de un predicado equipo/1 cuyo argumento es una constante que identifica el equipo. Los jugadores lo están en un predicado jugador/1 cuyo argumento es una constante que identifica el jugador. Cada hecho representa la respuesta de un encuestado.

Definir el predicado **equipos_preferidos/1** que verifica que su argumento es la lista de los equipos que aparecen el mayor número de veces en la encuesta.

Ejemplo:

?- equipos_preferidos(L).

L= [realmadrid,atletico]

con la encuesta:

equipo(realmadrid).	equipo(realmadrid).	equipo(atletico).	equipo(zaragoza).	equipo(barsa).
equipo(atletico).	equipo(realmadrid).	equipo(barsa).	equipo(atletico).	

Definir el predicado **jugadores_no_populares/1** que verifica que su argumento es la lista de los jugadores que aparecen el menor número de veces en la encuesta.

Ejemplo:

?- jugadores_no_populares(L).

L= [messi, benzema]

con la encuesta:

jugador(messi).	jugador(alves).	jugador(alves).	jugador(ronaldo).	jugador(costa).
jugador(benzema).	jugador(courtois).	jugador(ronaldo).	jugador(courtois).	jugador(costa).
jugador(ronaldo).	jugador(courtois).			

Nota: la ordenación de la lista es indiferente.

Nota: Las encuestas (los hechos de los predicados jugador/1 y equipo/1) deben figurar en un fichero independiente ("encuesta.pl") que se debe incluir en el programa. Es decir, los hechos no deben aparecer explícitamente en el programa.

PARTE 2: La bandera de Italia

Representaremos figuras geométricas por estructuras de dos argumentos, cuyo nombre es el de la figura, el primer argumento sus dimensiones y el segundo su color. Para las dimensiones se utiliza otra estructura, de nombre desconocido (pero siempre el mismo para cada tipo de figura) y tantos argumentos como sea necesario para definir las dimensiones de la figura. Así, tenemos estructuras de la forma X(D,C) donde X puede ser cuadrado, rectángulo, triangulo, etc.; D será otra estructura con un argumento que contiene la longitud del lado (para el cuadrado), dos argumentos con la base y la altura (para el rectángulo), tres argumentos con las longitudes de dos lados y el valor del ángulo que forman (para el triangulo), etc.; y C contendrá una constante que identifica un color: rojo, blanco, verde, azul, etc.

Definir el predicado **bandera_italia/2** que dada una lista de figuras geométricas L1 (que vendrá como entrada a la llamada en el primer argumento), las cuales pueden ser blancas, rojas o verdes, se verifica que el segundo argumento L2 contiene las figuras de L1 ordenadas como la bandera de Italia (es decir, primero las verdes, segundo las blancas y tercero las rojas).

Ejemplo:

?- bandera_italia([cuadrado(lado(5),rojo), rectangulo(bxh(3,4),rojo), triangulo(dosl(3,3,65),blanco), cuadrado(lado(4),blanco), rectangulo(bxh(2,3),verde)], L).

L= [rectangulo(bxh(2,3),verde), triangulo(dosl(3,3,65),blanco), cuadrado(lado(4),blanco), cuadrado(lado(5),rojo), rectangulo(bxh(3,4),rojo)]

Nota: la ordenación de las figuras del mismo color entre sí es indiferente. Puede ser la del ejemplo u otra cualquiera. No es necesario obtener todas las soluciones de ordenación diferente, solo una cualquiera.

Suponer que el usuario puede introducir (vía teclado) su figura geométrica favorita y la dimensión máxima de la misma, definir el predicado **bandera_italia_adhoc/2** que dada una lista de figuras geométricas L1 (blancas, rojas, o verdes) en el primer argumento, lee la figura geométrica favorita y la dimensión máxima (una después de la otra) y verifica que el segundo argumento L2 contiene las figuras de L1 que coinciden con la figura favorita y que tienen una dimensión menor o igual que la máxima ordenadas como la bandera de Italia.

Ejemplo:

?- bandera_italia_adhoc([rectangulo(bxh(2,3),verde), triangulo(dosl(3,3,65),blanco), cuadrado(lado(4),blanco), cuadrado(lado(5),rojo), rectangulo(bxh(3,4),rojo), cuadrado(lado(7),verde), triangulo(dosl(2,2,65),verde), cuadrado(lado(3),verde)],L).

|: cuadrado.

|: lado(6).

L = [cuadrado(lado(3),verde), cuadrado(lado(4),blanco), cuadrado(lado(5),rojo)]

Nota: Para comparar dimensiones deben ser estructuras idénticas (mismo nombre y aridad) y se utiliza orden alfabético-lexicográfico: se comparan alfabéticamente el nombre con el nombre y recursivamente los pares de argumentos en la misma posición, recorriendo las estructuras en profundidad. Este orden puede dar resultados como que bxh(3,10) es menor que bxh(4,5) (ya que 3 es menor que 4), aunque corresponde a un rectángulo de mayor área.

PARTE 3: Ordenación por criterio libre

Se trata de hacer un programa de orden superior para la ordenación de una lista, de manera que el criterio de ordenación está sin especificar y se obtiene como argumento. Por tanto, el criterio va a ser un objetivo ejecutable que se pasa en uno de los argumentos de la llamada al predicado principal del programa y puede ser tan complejo como, por ejemplo, el que corresponde al orden descendente por la longitud de cada elemento (dando por supuesto que son listas). Se debe:

1. Programar un predicado **qsort**(List,Order,Set) tal que Set es la lista List ordenada según el criterio Order. Utilizar el algoritmo de ordenación rápida (*quicksort* en las transparencias).

2. Escribir el segundo argumento de la llamada a **qsort/3** que correspondería al criterio habitual: el de ordenación ascendente por el valor numérico de los elementos.

Pista: El criterio es una relación entre cada dos elementos sucesivos de la lista ordenada, por lo que es necesario especificar una forma en la que identificar, del objetivo ejecutable que corresponde al criterio, los argumentos que corresponden a dichos elementos. Por ejemplo, para el criterio habitual el objetivo sería X=<Y, siendo X e Y elementos sucesivos de la lista; el programa debe ser capaz de identificar cual es cual: en este caso, Y es el que sigue a X (de lo contrario sería orden descendente).

Otros criterios que podrían considerarse son:

- Orden ascendente alfabético-lexicográfico del tercer argumento de los elementos de la lista.
- Orden descendente del valor numérico resultante del polinomio en dos variables de cada elemento de la lista con los valores de las variables a 2 y 3, respectivamente.
- Orden ascendente del valor resultante de sumar todos los nodos hoja de los árboles binarios elementos de la lista.
- Cualquier orden ascendente o descendente que se puede expresar como un objetivo Prolog.

Aclaración: Por ejemplo, podemos pensar en la siguiente llamada

qsort(L,"ascendente por el valor de la suma de las hojas de árboles binarios",S)

que debería ordenar ascendentemente por los valores de la suma de las hojas de los árboles binarios (es decir, se deben sumar las hojas de los elementos de la lista, que se suponen árboles binarios, y comparar para realizar la ordenación). En esta llamada, el criterio tal como aparece no está expresado como objetivo Prolog. El criterio tendrá que estar expresado en Prolog. Por tanto, se trata de definir qué forma debe tener el argumento del criterio y modificar *quicksort* como sea necesario para realizar las comparaciones correspondientes. No hay que programar ni "reunir las hojas de un árbol binario", ni "sumar los elementos de una lista", ni "calcular la longitud de una lista", ni ninguna parte del criterio (ya que no es posible programar todo lo que podría usarse como criterio).