Práctica 8:

COMBINATORIA

OBJETIVO

- Solución de problemas combinatorios en Prolog

TRABAJO PREVIO

Resuelve el **ejercicio a)** usando la explicación que tienes a continuación. Realiza una traza del predicado **perm/1** hasta el segundo resultado para entender cómo funciona. ¿Se podrían reordenar las instrucciones para que fuese más eficiente?

El previo se entregará por Aula Virtual y deberá incluir la traza como comentario.

PLANTEAMIENTO DEL PROBLEMA

El Prolog es un lenguaje muy adecuado para resolver problemas de combinatoria utilizando **backtracking**. Mediante hechos se representan los posibles valores que podemos seleccionar. Por ejemplo, si queremos obtener todas las permutaciones posibles con 3 elementos (0, 1, 2), representaremos estos valores mediante hechos:

```
num(0).
num(1).
num(2).
```

El predicado para solucionar el problema deberá consultar estos hechos para cada una de las variables de nuestro problema y a continuación colocar las restricciones que deben cumplir estas variables, de manera que si no se cumplen, el Prolog hará backtracking y probará con otra combinación.

El predicado para solucionar este problema sería:

```
perm([X1, X2, X3]):-
  num(X1),
  num(X2),
  num(X3),

  % Restricciones que deben cumplir: ser distintas entre si
  X1 \== X2,
  X1 \== X3,
  X2 \== X3.
```

Con este programa obtendríamos una solución al problema, es decir, una permutación. Para obtenerlas todas, se puede utilizar el predicado *findall*, que obtiene todas las soluciones de un predicado y las guarda en una lista. A continuación usaríamos el predicado *length* para saber cuántas soluciones hay en la lista. Aquí se muestra el predicado que resolvería el ejercicio a)

```
ejer_a(N):-
  findall(Sol, perm(Sol), L), length(L,N).
```

Utilizar Prolog para resolver los siguientes problemas de combinatoria. **Comprueba** si los resultados son correctos resolviéndolos también en papel.

a) ¿Cuántas permutaciones hay de 3 elementos? (Consiste en escribir y probar lo comentado más arriba)

b) ¿Cuántas soluciones de números naturales (incluido el cero) existen para X1+X2+X3+X4 = 10 ? Ayuda: cada solución es equivalente a una forma de colocar 10 bolas en 4 cajas.

c) Tenemos un juego en el que hay que llevar una canica desde el inicio hasta la meta a través de un laberinto con 6 agujeros a esquivar. Cada vez que la canica se cae por un agujero aparece justo después del agujero, pero si se cae más de 2 veces, hemos perdido. ¿Cuantas formas distintas de llegar hasta la meta tenemos?

d) ¿Cuantas manos de poker distintas hay? Los 4 palos de la baraja los representaremos como: *p, c, r, t*. Los valores de las cartas por los números del 1 al 10 más las letras j, q, k. Una carta la representaremos mediante la función *c* con 2 parámetros, por ejemplo, el rey de picas sería: c(k, p). Ayuda: una mano de poker está formada por 5 cartas diferentes de la baraja francesa.

e) Dada una función con un parámetro de entrada de tipo vector:

```
typedef unsigned char Vector[N];
void f(Vector v);
```

a) Encontrar cuantos vectores diferentes puede haber suponiendo que los enteros solo pueden ser valores del 0 al 7, y que N=3. Este problema y los siguientes se deberán resolver mediante un **predicado recursivo** que permita resolver el problema de forma general para vectores de cualquier tamaño.

b) Encontrar cuántos de los vectores anteriores comienzan por el número 0

c) Encontrar cuántos de los vectores anteriores comienzan por su valor mínimo

d) (Opcional) Repetir los apartados anteriores en el caso que no se permitan valores repetidos en un vector.