Práctica 6:

LABERINTO

OBJETIVO

- Representación de grafos en Prolog.
- Recorrido de un grafo.

TRABAJO PRELIMINAR:

Leer y entender el enunciado. Escribir el grafo del apartado 'a' en un fichero, representándolo mediante hechos **arco/3** tal como se explica a continuación.

REPRESENTACIÓN DE GRAFOS

En esta práctica se debe implementar un predicado que nos permita encontrar un camino dentro de un laberinto desde un punto cualquiera a otro. Para ello representaremos el laberinto mediante un grafo. En este primer apartado usaremos un grafo dirigido, puesto que resulta más sencillo, y en el siguiente usaremos grafos no dirigidos.

Un grafo se representa en Prolog utilizando hechos, donde cada hecho representa un arco del grafo. Cada hecho tendrá tres parámetros: nodo origen del arco, nodo destino del arco y longitud del arco.

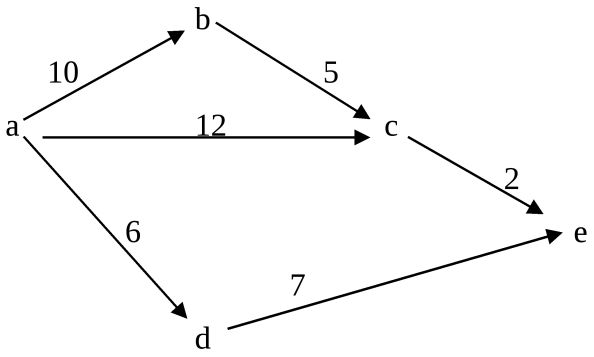
Ej.

```
arco(a, b, 10).
arco(b, c, 5).
arco(a, c, 12).
```

Problema

a) Implementar el predicado *camino/4*. A este predicado se le pasan como parámetros de entrada el nodo de inicio y el nodo final, y tendrá como resultados el camino y su longitud. El camino será una lista con todos los nodos del camino, incluyendo el nodo inicial y el final.

El predicado camino se deberá probar sobre el siguiente grafo:



b) En un laberinto real, cada tramo del laberinto se puede recorrer en las dos direcciones: hacia delante y hacia atrás. El grafo correcto para representar este laberinto será, por tanto, un grafo no dirigido. Para conseguir que nuestro laberinto sea un grafo no dirigido se puede definir un nuevo tipo de arco:

```
arcond(X, Y, Dist):-
   arco(X, Y, Dist);
   arco(Y, X, Dist).
```

Escribir un nuevo predicado *caminond/5* similar al predicado *camino/4* para que sea capaz de recorrer un grafo no dirigido.

Para evitar que el algoritmo se quede en ciclos infinitos, hay que llevar la cuenta de los nodos que hemos recorrido para no pasar dos veces por el mismo sitio. Los nodos recorridos se guardarán en una lista que corresponderá con el quinto parámetro. El algoritmo, cada vez que pase por un nodo nuevo, deberá comprobar que no está en la lista. Para ello se utilizará el predicado *member/2*, que es equivalente al predicado *miembro/2* que definisteis en una práctica anterior.

c) Realizar un predicado *minimo/4* que calcule el camino mínimo y su longitud (los datos de entrada serán el nodo de inicio y el nodo final), utilizando para ello el predicado *caminond/5*. Para realizar este predicado tener en cuenta la siguiente definición de mínimo para un conjunto de números:

$\forall X$ (minimo(X) sii no ($\exists Y$ Y < X))