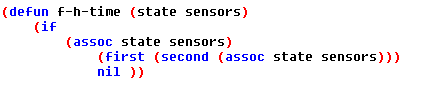
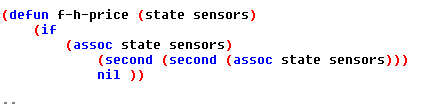
***Inteligencia Artificial: Práctica 2***

***Ejercicio 1:***

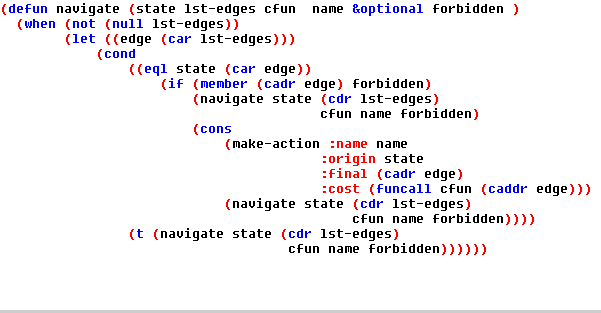
Para el ejercicio, hemos desarrollado dos funciones que calculan los valores de las dos posibles heurísticas:

1. **F-h-time**: Si en la lista de heurísticas está el estado pedido, devolvemos el primer elemento de la segunda sublista (la que contiene los valores), que es el que corresponde al tiempo.
2. **F-h-price**: Si en la lista de heurísticas está el estado pedido, devolvemos el segundo elemento de la segunda sublista (la que contiene los valores), que es el que corresponde al coste.



***Ejercicio 2:***

En este segundo ejercicio, se nos pide hacer una función que devuelva las posibles acciones desde un nodo, dado el estado (nombre del nodo). Para ello, hemos desarrollado una función general, y seguidamente una serie de funciones secundarias que, depende de lo pedido, llaman a esta primera con distintos argumentos. Las funciones son las siguientes:

1. **Navigate**: Función general, que recibe como argumentos un estado, la lista de conexiones del grafo, la función para acceder al valor buscado (en el caso del ejemplo, tiempo o precio), el nombre a dar a las acciones generadas y opcionalmente, una lista de ciudades a las que no se puede llegar en tren. Esta función recorre los elementos de la lista de conexiones, comprobando si el origen de cada conexión es el estado pasado como argumento. En tal caso, si el destino no está en la lista de prohibidos, crea una acción con los campos correspondientes, y sigue recorriendo las conexiones. En cualquier otro caso, ignora esa conexión y sigue recorriendo.
2. **Navigate-canal-time**: Obtiene las posibles acciones por canales y según el tiempo desde un estado dado. Usa Navigate.



1. **Navigate-canal-price**: Obtiene las posibles acciones por canales y según el precio desde un estado dado. Usa Navigate.



1. **Navigate-train-time:** Obtiene las posibles acciones en tren y según el tiempo desde un estado dado. Usa Navigate.

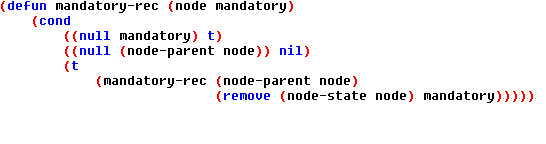


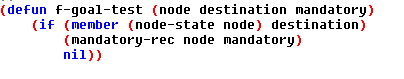
1. **Navigate-train-price:** Obtiene las posibles acciones en tren y según el precio desde un estado dado. Usa Navigate.



***Ejercicio 3:***

Se nos pide desarrollar una función que compruebe si se ha alcanzado el objetivo. Para ello, hay que comprobar que el estado del nodo dado es un estado final, y que ha pasado por todas las ciudades obligatorias. Para lo segundo, hemos creado una función auxiliar. En total tenemos estas dos funciones:

1. **Mandatory-rec:** Va recorriendo los antecesores del nodo dado, comprobando en cada iteración si la ciudad visitada está en la lista de obligatorias y, en tal caso, eliminándolas de esta. La función devolverá si se han visitado todas o si se ha llegado al final y no ha pasado por todas.
2. **F-goal-test:** Si el estado del nodo es un estado final, devuelve lo que devuelva Mandatory-rec, en otro caso devuelve nil.



***Ejercicio 4:***