

Taller 6





























Luis Felipe Garzón Bonilla 1089931169

Steven Grisales López 1034289634

En la documentación del código en prolog se puede observar el planteamiento de cada ejercicio y su respectivo desarrollo con explicación.

```
1 /* Listas */
2 % [cabeza, cola]      [] -> lista
3
4 /* Hallar el último elemento de una lista. */
5 ultimo([Result], Result).                % Base
6 ultimo([], Result) :- Result = "La lista esta vacia". % Caso para cuando la lista esta vacia
7 ultimo([_|L], Result) :- ultimo(L, Result). % Recursividad
8
9 % Consultas = ultimo([a, [b,c], 2], Ultimo). || ultimo([], Ultimo).
10
11
12 /* Hallar el elemento k de una lista.*/
13 elemento_k([Result|_], 0, Result).        % Base
14 elemento_k([_|L], K, Result) :- K > 0,    % Recursividad
15                                     K1 is K-1,
16                                     elemento_k(L, K1, Result).
17
18 % Consultas = elemento_k([a, [b,c], 2], 2, Elemento).
19
20 /* Método member(?Elem, ?List) True if Elem is a member of List. */
21 miembro([X|_], X).
22 % La condición de parada será cuando el elemento a buscar sea encontrado en la cabeza
23
24 miembro([_|L], X) :- miembro(L, X).
25 % Se examina cada elemento de la lista volviendo a llamar a miembro pero unificando L con [X|_]
26
27 % Consulta = miembro([6,4,2,7],4).
28
29 /* Método proper_Length(@List, -Length) True when Length is the number of elements in the proper List List.*/
30 longitud([],0).
31 % La condición de parada será cuando la lista este vacia, por lo tanto su longitud es de 0.
32
33 longitud([_|T], L) :- longitud(T, L1), L is L1 + 1.
34 % Ya que en listas se va descartando la cabeza, vamos contando los elementos que van pasando por la cola.
35
36 % Consulta = longitud([6,4,2,7],4).
37
38 % trace es para mostrar el debug y se puede usar para entender la recursividad de los códigos.
39
40 /* Información: https://swish.swi-prolog.org/p/Tutorial%20de%20prolog.swinb
41 https://www.swi-prolog.org/pldoc/man?section=lists */
```

Consultas Realizadas:

 ultimo([a, [b,c], 2], Ultimo).	  
Ultimo = 2	
Ultimo = "La lista esta vacia"	
 ultimo([], Ultimo).	  
Ultimo = "La lista esta vacia"	
 elemento_k([a, [b,c], 2], 2, Elemento).	  
Elemento = 2	
false	
 miembro([6,4,2,7],4).	  
true	1
false	
 miembro([6,2,2,7],4).	  
false	
 longitud([6,4,2,7],4).	  
true	1
 longitud([6,4,2,7],7).	  
false	