**TC2006 – Lenguajes de Programación***Ejecución y Programación Recursiva en Prolog*

Matrícula1\_\_\_\_\_\_\_\_\_\_ Matrícula2\_\_\_\_\_\_\_\_\_\_\_

1. De forma similar a lo visto en clase, ilustra el árbol de búsqueda que se formará con el siguiente programa en Prolog al ejecutarlo para obtener todas las respuestas para los querys: (a) **?- a(Z).** (b) **?- a(4).**

a(X1) :- b(X1). % #1

a(X2) :- f(X2). % #2

b(X3) :- g(X3), !, v(X3). % #3

b(X4) :- X4 = 4, v(X4). % #4

g(1). % #5

g(2). % #6

g(3). % #7

v(X5). % #8

f(5). % #9

**Programar los siguientes predicados en prolog usando el cut para que siempre regrese un solo resultado.**

1. Programar el predicado **bicond** que implemente el bicondicional (<=>).

Probar con:

?- bicond(true,true). **=> true.**

?- bicond(true,false). **=> false.**

?- bicond(2 < 3,7 > 2). **=> true.**

?- bicond(2 > 3,7 > 2). **=> false.**

1. Programar el predicado **internos** que cuente los nodos internos de un árbol binario (que tiene al menos un subárbol no vacío) descrito con la función: **arbol(Raíz, SubárbolIzquierdo, SubárbolDerecho)**.

Probar con:

?- internos(arbol(8,arbol(5,arbol(2,nil,nil),arbol(7,nil,nil)),

arbol(9, nil, arbol(15, arbol(11, nil, nil), nil))),I). **=> I = 4.**

1. Programar el predicado **decrementa** que busque un elemento, que debe pedirle al usuario, en un árbol binario y que si lo encuentra, le reste una cantidad solicitada; de lo contrario que regrese el árbol original.

Probar con:

?- decrementa(arbol(8,arbol(5,arbol(10,nil,nil),arbol(7,nil,nil)),

arbol(9, nil, arbol(15, arbol(11, nil, nil), nil))), A).

**Elemento? 10**

**Decremento? 4**

**=> A = arbol(8,arbol(5,arbol(6,nil,nil),arbol(7,nil,nil)),**

**arbol(9, nil, arbol(15, arbol(11, nil, nil), nil)))**

?- decrementa(arbol(8,arbol(5,arbol(2,nil,nil),arbol(7,nil,nil)),

nil, A).

**Elemento? 1**

**No se encuentra en el árbol**

**=> A = arbol(8,arbol(5,arbol(2,nil,nil),arbol(7,nil,nil)), nil).**