Ejercicio 1 *Se considera el siguiente programa lógico*

```
n(0).
n(s(X)) :-
    n(X).

m(0,s(Y)) :-
    n(Y).
m(s(X),s(Y)) :-
    m(X,Y).
```

Dibujar los árboles de resolución correspondiente al programa y a las siguientes preguntas, indicando las respuestas obtenidas.

1. `?- m(X,s(s(0))).`

2. `?- m(X,s(X)).`

Solución:
El árbol correspondiente a la primera pregunta se representa en la figura 1 Las respuestas son

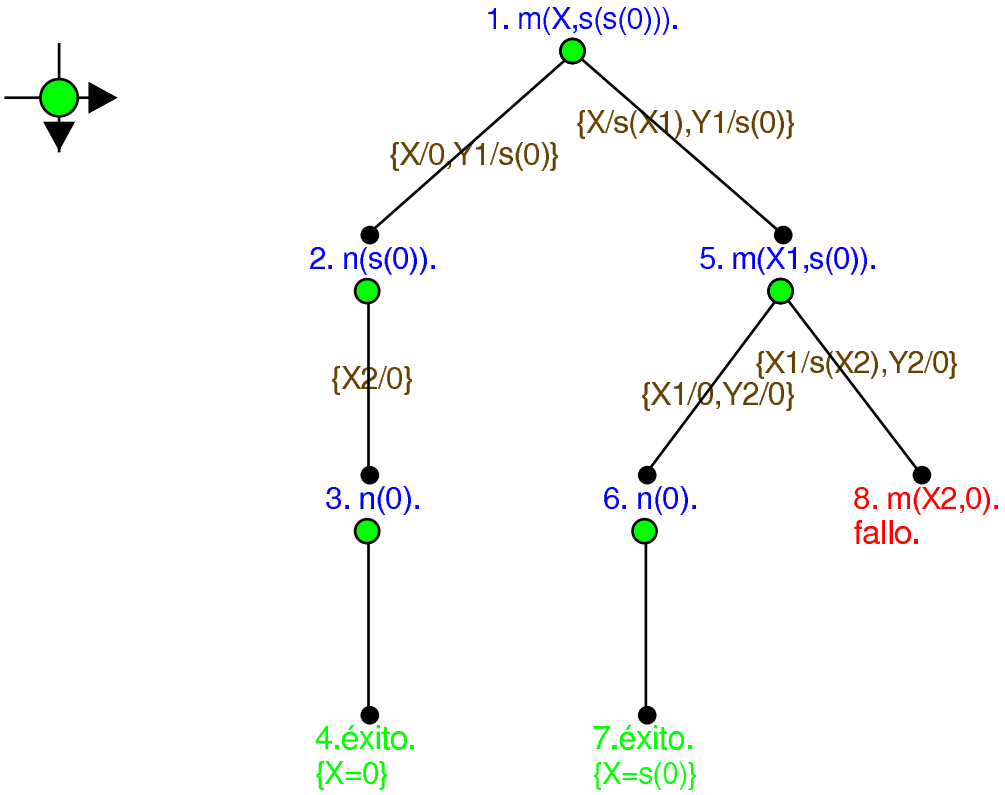

Figura 1: Pregunta 1

`0` y `s(0)`.

El árbol correspondiente a la segunda pregunta se representa en la figura 2 Hay infinitas respuestas: `0`, `s(0)`, `s(s(0))`, ...

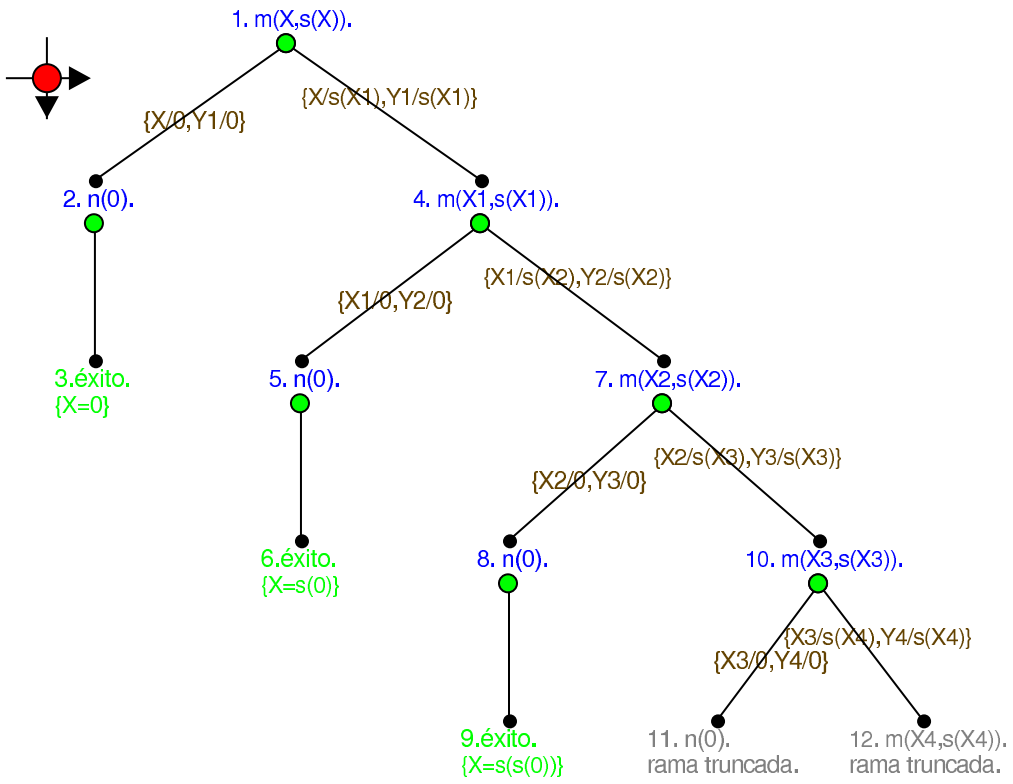

Figura 2: Pregunta 2

Ejercicio 2 *Se desea definir la relación* `decreciente(+N,-L)` *que se verifique si* `L` *es la lista de los números decrecientes de* `N` *hasta* `1`*. Por ejemplo,*

```
?- decreciente_4(5,L).
L = [5, 4, 3, 2, 1]
```

Determinar cuáles de las siguientes definiciones son correctas y, en las incorrectas, explicar el comportamiento de Prolog con la pregunta

```
?- decreciente(5,L).
```

1. *Primera definición*

```
decreciente(1,[1]).
decreciente(N,[N|L]) :-
    N > 1,
    N1 is N-1,
    decreciente(N1,L).
```

2. *Segunda definición*

```
decreciente(1,[1]).
decreciente(N,[N-1|L]) :-
    N > 1,
    N1 is N-1,
    decreciente(N1,L).
```

3. *Tercera definición*

```
decreciente(1,[1]).
decreciente(N,[N|L]) :-
   N > 1,
   decreciente(N1,L),
   N1 is N-1.
```

4. *Cuarta definición*

```
decreciente(N,L) :-
   findall(X,(between(1,N,Y), X is N+1-Y),L).
```

Solución:
La primera definición es correcta.
La segunda es incorrecta. La respuesta a la pregunta es `L = [5-1, 4-1, 3-1, 2-1, 1]`.
La segunda es incorrecta. La respuesta a la pregunta es un mensaje de error al hacer `N > 1` la variable `N` no tiene valor.
La cuarta definición es correcta.

Ejercicio 3 *Definir la relación* `únicos(+L1,-L2)` *que se verifique si* `L2` *es la lista de los elementos que ocurren únicamente una vez en la lista* `L1`*. Por ejemplo,*

```
?- únicos([2,5,3,2],L).
L = [5, 3]
?- únicos([2,5,5,2],L).
L = []
```

Solución:
La definición de `únicos` es

```
únicos(L1,L2) :-
   findall(X,es_único(X,L1),L2).
```

donde la relación `es_único(?X,+L)` se verifica si `X` es un elemento que ocurre únicamente una vez en la lista `L`. Por ejemplo,

```
?- es_único(X,[2,5,3,2]).
X = 5 ;
X = 3 ;
No
```

```
es_único(X,L) :-
   select(X,L,R),
   not(memberchk(X,R)).
```

Ejercicio 4 *Definir la relación* `transforma(+L1,-L2)` *que se verifique si* `L2` *es la lista obtenida añadiéndole a cada elemento numérico de* `L1` *su posición en la lista. Por ejemplo,*

```
?- transforma([1,1,1,a,b,c,1,1,1],L).
L = [2, 3, 4, a, b, c, 8, 9, 10]
?- transforma([1,2,a,5,2,b,3,1],L).
L = [2, 4, a, 9, 7, b, 10, 9]
```

Dar dos definiciones, una recursiva y otra no–recursiva.

Solución:
La definición de `transforma` es

```
transforma(L1,L2) :-
   transforma_aux(L1,1,L2).
```

donde `transforma_aux(+L1,+N,-L2)` se verifica si `L2` es la lista obtenida añadiéndole a cada elemento numérico de `L1` la suma de `N` y su posición en la lista. Por ejemplo,

```
transforma_aux([],_,[]).
transforma_aux([X|L1],N,[Y|L2]) :-
   number(X), !,
   Y is X+N,
   N1 is N+1,
   transforma_aux(L1,N1,L2).
transforma_aux([X|L1],N,[X|L2]) :-
   % not(number(X)),
   N1 is N+1,
   transforma_aux(L1,N1,L2).
```

La definición no recursiva de `transforma` es

```
transforma_2(L1,L2) :-
   lista_de_posiciones(L1,L),
   maplist(suma,L1,L,L2).
```

donde `lista_de_posiciones(+L1,-L2)` se verifica si `L2` es la lista de posiciones correspondiente a la lista `L1`. Por ejemplo,

```
?- lista_de_posiciones([1,1,1,a,b,c,1,1,1],L).
L = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
lista_de_posiciones(L1,L2) :-
   length(L1,N),
   findall(X,between(1,N,X),L2).
```

y `suma(+X,+Y,-Z)` se verifica si `Z` es la suma de `X` y el número `Y`, cuando `X` es un número y es igual a `X`, en caso contrario. Por ejemplo,

```
?- suma(3,2,Z).
Z = 5
?- suma(b,2,Z).
Z = b
```

```
suma(X,Y,Z) :-
   number(X), !,
   Z is X+Y.
suma(X,_,X).
```