Solución

Ejercicio 2

a)

Soluciones:

Ys = [1, 3]
Ys = [2, 3]
Ys = []

Justificación:

? r([1,2,3], f(f(a)), Ys)

- r([2, 3], f(a), _G321) Ys = [1|_G321]
  - r([3], a, _G322) _G321 = [2|_G322]
    - No unifica.
  - r([3], f(a), _G321)
    - r([], a, _G323) _G321 = [3]
      - **Unifica únicamente con la primera cláusula: _G323 = [] y, por lo tanto, Ys = [1,3]**
    - r([], f(a), _G321)
      - **No unifica.**
- r([2, 3], f(f(a)), Ys)
  - r([3], f(a), _G324) Ys = [2|_G324]
    - r([], a, _G325) _G324 = [3|_G325]
      - **Unifica únicamente con la primera cláusula: _G325 = [] y, por lo tanto, Ys = [2,3]**
    - r([], f(a), _G324)
      - **No unifica.**
  - r([3], f(f(a)), Ys)
    - r([], f(a), _G326) Ys = [3|_G326]
      - **No unifica.**
    - r([], f(f(a)), Ys)
      - **Unifica únicamente con la primera cláusula: Y = []**

b)

```
i.      r([],N,[])        :-    p(N), !.
ii.     r([X|Xs],f(N),[X|Ys])   :-    !, r(Xs,N,Ys).
iii.    r([X|Xs],f(N),[X|Ys])   :-    r(Xs,N,Ys), !.
```

c)

No se modifica, porque el segundo argumento de r/3 está instanciado. Esto se debe a que en última instancia, un cut en la primera cláusula de p/1 solo va a evaluarse en las hojas del árbol de búsqueda.

Sin embargo, si el segundo argumento no está instanciado la consulta sí cambia. Por ejemplo:

? r([1,2,3], N, Ys).

Ejercicio 3

a) Implemente el siguiente metaintérprete para *Prolog* puro:

```
resLim((true,TG), N) :-
        !,
        resLim(TG, N).
resLim((HG,TG), N) :-
        !,
```

```
        largo(TG, LTG),
        LTG < N,
        clause(HG, B),
        juntar(B, TG, NG),
        resLim(NG, N).
resLim(true, _) :-
        !.
resLim(G, N) :-
        !,
        0 < N,
        clause(G, B),
        resLim(B, N).

largo((_,B), N) :-
        !,
        largo(B, N1),
        N is N1+1.
largo(_, 1).

juntar((H,T), O, Res) :- % Res no se unifica en la clausula por el cut
        !,
        juntar(T, O, TO),
        Res = (H, TO).
juntar(L, O, (L,O)).
```

b)

```
    % natural (-N)<- N es un natural
    natural(0).
    natural(N):-
        natural(N1),
        N is N1+1.

    % entre(+I, +J, ?K)<- K es un natural entre I y J
    entre(I,J,I):-
        I=<J.

    entre(I,J,K):-
        I<J,
        I1 is I+1,
        entre(I1,J,K).

    % S <- genera una tira de a*b*
    s -->
        {natural(N),
         entre(0,N,M),
         J is N-M},
        a(J),
        b(M).

    % a(N) <- genera la tira a^N
    a(0)--> [].

    a(N)-->
        {N>0,
        N1 is N-1},
        [a],
        a(N1).

    % b(N) <- genera una tira de la forma b^N
    b(0)--> [].

    b(N)-->
        {N>0,
         N1 is N-1},
         [b],
         b(N1).
```

Ejercicio 4

a) Dé una interpretación con dominio natural, que no sea modelo de P.

f y g son la función identidad, y q y p son el predicado que nunca se satisface. Como hay hechos con q y p, estos no van a satisfacerse en la interpretación lo que la hace que no sea modelo.

b) Defina respuesta computada y respuesta correcta.

Del teórico:

Una respuesta computada para $P \cup \{G\}$ es la sustitución obtenida de la composición de mgus utilizados en una refutación SLD de $P \cup \{G\}$ restringiéndola a las variables de G

Sea P un programa lógico y G = <- $A_1 \dots A_k$ un objetivo definido.
Una respuesta correcta para $P \cup \{G\}$ es una respuesta ϕ que cumple:
$P \models \forall((A_1 \wedge \dots \wedge A_k)\phi)$

c) Construya los árboles SLD correspondientes a $P \cup \{G_1\}$.

	:- q(a,A), p(f(B),A)
1	:- q(a,b)
2	:- p(a,a)
3	Solución {A/b, B/a}
4	:- q(a,c)
5	:- q(a,Z), p(g(Z), c)
6	:- q(a,Z), q(c,Z)
7	:- q(a,f(c)) Falla
8	:- q(a,b), p(c,c) Falla
9	:- q(a,c), q(c,Z_1), p(g(Z_1), c)
10	:- q(a,c), q(c,Z_1), q(c,Z_1)
11	:- q(a,c), q(c,f(c)), q(c,f(c))
12	:- q(a,c), q(c,f(c))
13	:- q(a,c) ...
14	:- q(a,c), q(c,b), p(c,c) Falla
15	:- q(a,c), q(c,c), q(c, z2), p(g(Z_2),c)
16	:- q(a,c), q(c,c), q(c, z2), q(c,Z2)

Las resoluciones hechas son, para cada línea:

1. p(f(a), b).
   {B/a, A/b}
2. q(X, b) :- p(X, X).
   {X/a}
3. p(a, a).
4. p(f(a), c).
   {B/a, A/c}
5. q(X, c) :- q(X, Z), p(g(Z), c).
   {X/a}
6. p(g(Y),X):- q(X,Y).
   {Y/Z. X/c}
7. q(X, f(X)).
   {Z/f(c)}
8. q(X, b) :- p(X, X).
   {Z/b}
9. q(X, c) :- q(X, Z_1), p(g(Z_1), c).
   {X/c}
10. p(g(Y),X):- q(X,Y).
    {Y/Z_1, X/c}

Vemos que la línea 10 tiene como átomo de más a la derecha a g(c, Z_1), esencialmente lo mismo que la línea 6. Entonces hay una rama infinita sin soluciones.

d) Dé las soluciones que muestra el intérprete, si éste selecciona las cláusulas en el orden de aparición en el programa y realiza una búsqueda DFS.

La única solución es {A/b, B/a}.

e) ¿Existe alguna respuesta correcta que no sea computada? Justifique.

No, ya que la respuesta es ground.

Ejercicio 5

```
%simbolo(?N) <- N es un símbolo válido para clave.
simbolo(rojo).
simbolo(verde).
simbolo(azul).
simbolo(blanco).
simbolo(negro).
simbolo(violeta).


%largo_clave(N) <- N es el largo de la clave
largo_clave(4).

%alfabeto(-Alfa) <- Alfabeto es el conjunto de símbolos para utilizar en claves.
alfabeto(Alfa):-
        setof(Sim, simbolo(Sim), Alfa).


%clave_maestro(-Clave) <- Clave es una clave generada al azar.
clave_maestro(Clave):-
        largo_clave(N),
        alfabeto(Alfa),
        largo(Alfa,M),
        clave_maestro(Clave,N,Alfa-M).


%clave_maestro(-Clave,+N, +Alfa-M) <- Clave es una clave de N símbolos generada al azar
                                     sobre el alfabeto Alfa de largo M
clave_maestro([],0,_):-!.
clave_maestro([Num|Resto], N,Alfa-M):-
        N>0,
        !,
        J is random(M),
        elementoN(Alfa,J,Num)
        N1 is N-1,
        clave_maestro(Resto,N1,Alfa-M).

%elementoN(+Xs,+N,?X) <- X es el enésimo elemento de Xs.
elementoN([X|_],0,X):-!.
elementoN([_|Xs],N,X):-
        N>0,
        !,
        N1 is N-1,
        elementoN(Xs,N1,X).

%largo(+Xs,?N) <- Xs es una lista de largo N
largo([],0):-!.
largo([_|Xs],N):-
        largo(Xs,M),
        N is M+1.

% coincidencias(Prueba, Tot) <- Ver letra.
coincidencias(Prueba, Tot):-
        interac(Prueba, N),
        coinci(Prueba, N, Tot).

% coinci(Prueba, N, Tot) <- Prueba es una clave,  N es la cantidad ingresada,
    Tot es un valor válido.
coinci(Prueba, N, Tot):-
        N<0,
        !,
        writeln('¡Error! Debe ser al menos cero.'),
        nl,
        interac(Prueba,N1),
        coinci(Prueba,N1, Tot).
```

```
coinci(Prueba, N, Tot):-
        largo_clave(M),
        N>M,
        !,
        write('¡Error! No puede ser mayor a '),
        writeln(M),
        nl,
        interac(Prueba,N1),
        coinci(Prueba,N1,Tot).

coinci(_,Tot,Tot).


%interac(Prueba, N)<- Prueba es la clave a desplegar; N es el valor que se ingresa.
interac(Prueba, N):-
        write('La prueba es: '),
        writeln(Prueba),
        nl,
        write('¿Cuántas coincidencias hubo? '),
        nl,
        read(N),
        nl.
```