Prolog

Built-ins

- X is Y the value of X is unified with Y
- X =:= Y the values of X and Y are equal
- X =\= Y the values of X and Y are different
- X > Y the value of X is greater than the one of Y
- X < Y the value of X is lower than the one of Y
- X >= Y the value of X is greater than or equal to the one of Y
- X =< Y the value of X is lower than or equal to the one of Y

- X is Y the value of X is unified with Y
- X =:= Y the values of X and Y are equal
- X =\= Y the values of X and Y are different
- X > Y the value of X is greater than the one of Y
- X < Y the value of X is lower than the one of Y
- X >= Y the value of X is greater than or equal to the one of Y
- X =< Y the value of X is lower than or equal to the one of Y

- ! *Cut* predicate, allows us to prune useless paths

An example of the use of !

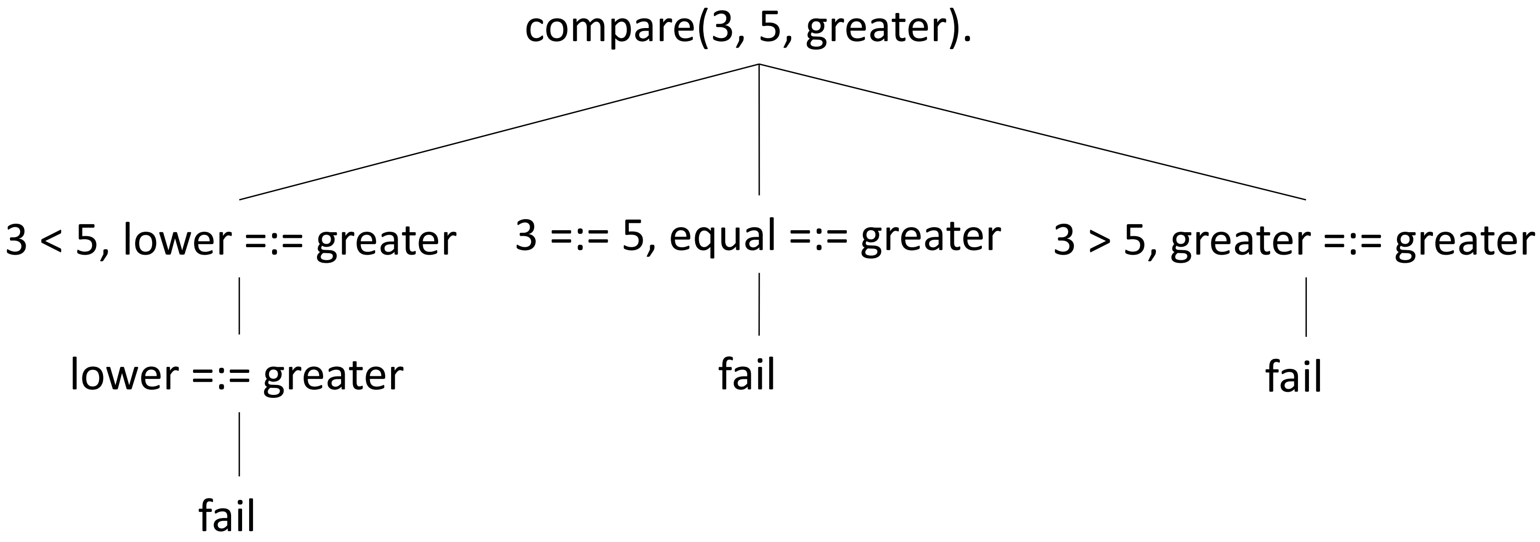
```
compare(X, Y, lower) :- X < Y.
compare(X, Y, equal) :- X =:= Y.
compare(X, Y, greater) :- X > Y.

?- compare(3, 5, greater).
```

An example of the use of !

```
compare(X, Y, lower) :- X < Y.
compare(X, Y, equal) :- X =:= Y.
compare(X, Y, greater) :- X > Y.
```

```
?- compare(3, 5, greater).
```



compare(3, 5, greater).

3 < 5, lower =:= greater

3 =:= 5, equal =:= greater

3 > 5, greater =:= greater

lower =:= greater

fail

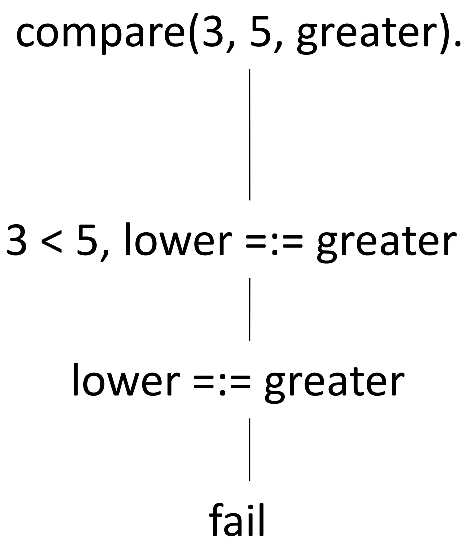
fail

fail

An example of the use of !

```
compare(X, Y, lower) :- X < Y , !.
compare(X, Y, equal) :- X =:= Y, !.
compare(X, Y, greater) :- X > Y.


?- compare(3, 5, greater).
```



compare(3, 5, greater).

3 < 5, lower =:= greater

lower =:= greater

fail

Prolog

Exercises

1) Compute the absolute value of a number

1) Compute the absolute value of a number

```
abs(X,X) :- X >= 0.
abs(X,Y) :- X < 0, Y is -X.
```

2) Compute the factorial of a number

2) Compute the factorial of a number

Intuitively:

fatt(0) = 1

fatt(n) = n * fatt(n-1) (per n>0)

2) Compute the factorial of a number

In Prolog:

```
fatt(0,1).
fatt(N,X) :- N>0, N1 is N-1, fatt(N1, X1), X is N*X1.
```

2) Compute the factorial of a number

A different Prolog version, using tail recursion:

```
fatt2(N,X) :- fatt2(N,1,X).
fatt2(0,ACC,ACC).
fatt2(M,ACC,X) :- ACC1 is M*ACC, M1 is M-1,
        fatt2(M1,ACC1,X).
```

3) Compute the greatest common divisor

3) Compute the greatest common divisor

Intuitively:

$\mathrm{GCD}(x,0) = x$

$\mathrm{GCD}(x,y) = \mathrm{GCD}(y, x \bmod y)$ (for $y>0$)

3) Compute the greatest common divisor

In Prolog:

```
gcd(X,0,X).
gcd(X,Y,Z) :- Y>0, X1 is X mod Y, gcd(Y,X1,Z).
```

4) Find the last element of a list

4) Find the last element of a list

```
last([X], X).
last([_|Z], X) :- last(Z,X).
```

4) Find the last element of a list

An alternative, using the built-in function *reverse:*

```
last(L, X) :- reverse(L, [X|_]).
```

5) Check if a list is a sublist of another list

5) Check if a list is a sublist of another list

```
sublist([], _).
sublist([X|L1], [X|L2]) :- sublist(L1, L2).
sublist([X|L1], [_|L2]) :- sublist([X|L1], L2).
```

Note that in this solution we are checking whether L1 is a sublist of L2.

6) Count how many times a number appears in a list

6) Count how many times a number appears in a list

```
count([], _, 0).
count([X|T], X, ACC) :- count(T, X, ACC2),
        ACC is ACC2 +1.
count([_|T], X, ACC) :- count(T, X, ACC).
```

7) Find the intersection between two sets

7) Find the intersection between two sets

For simplicity, we will assume that the lists do not contain repeated elements.

7) Find the intersection between two sets

```
intersection([], _, []).
intersection([H|T], SET2, [H|RES]) :-
        member(H, SET2),
        !,
        intersection(T, SET2, RES).
intersection([_|T], SET2, RES) :-
        intersection(T, SET2, RES).
```

8) Write your own *flatten* predicate.

8) Write your own *flatten* predicate.

Just in case…

The flatten predicate flattens a list. For example:

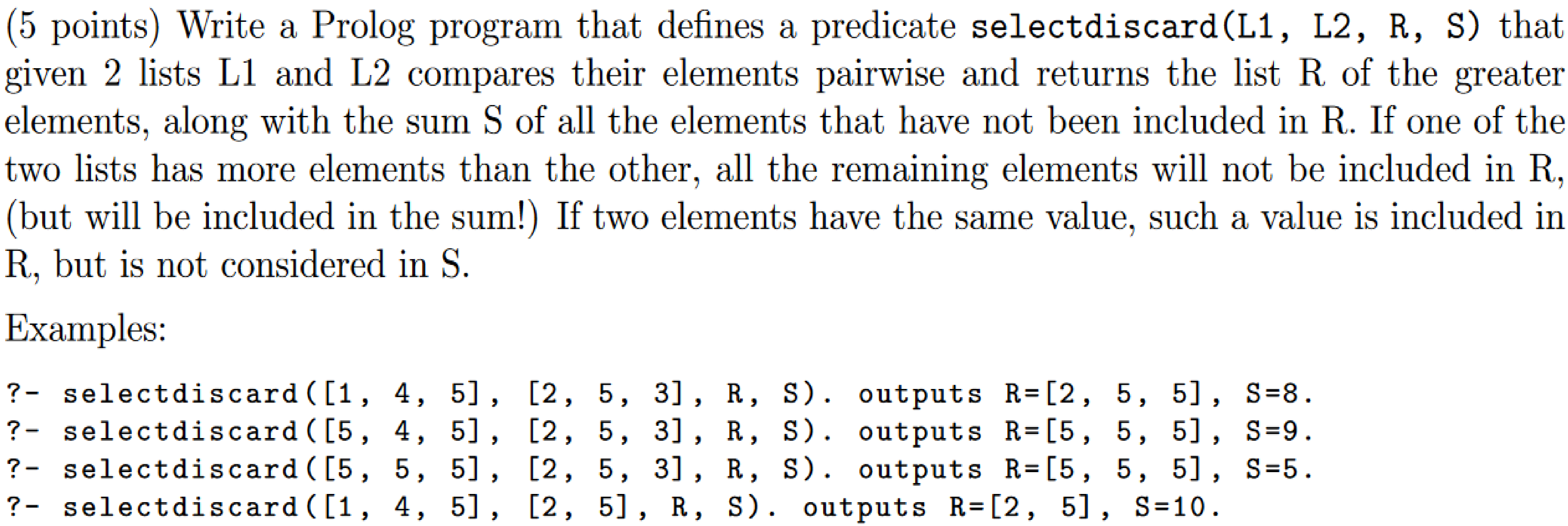
```
?- flatten([1, [2 , [3]], 4, [5, 6]], X).
X = [1, 2, 3, 4, 5, 6]
```

8) Write your own *flatten* predicate.

```
flatten2([], []) :- !.
flatten2([H|T], RES) :-
    !,
    flatten2(H, NewH),
    flatten2(T, NewT),
    append(NewH, NewT, RES).
flatten2(H, [H]).
```

Exercises from old exams:

Exercises from old exams:

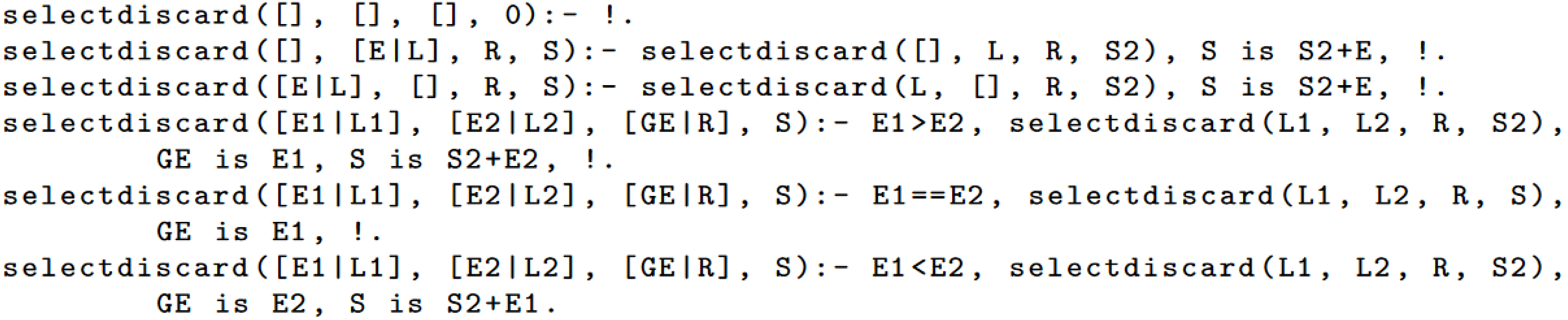


(5 points) Write a Prolog program that defines a predicate `selectdiscard(L1, L2, R, S)` that given 2 lists L1 and L2 compares their elements pairwise and returns the list R of the greater elements, along with the sum S of all the elements that have not been included in R. If one of the two lists has more elements than the other, all the remaining elements will not be included in R, (but will be included in the sum!) If two elements have the same value, such a value is included in R, but is not considered in S.

Examples:

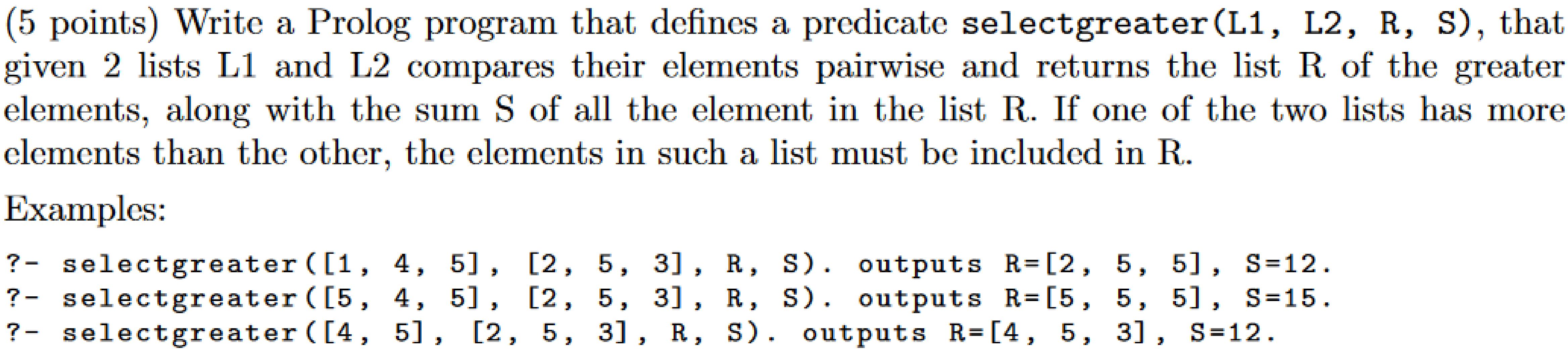
```
?- selectdiscard([1, 4, 5], [2, 5, 3], R, S). outputs R=[2, 5, 5], S=8.
?- selectdiscard([5, 4, 5], [2, 5, 3], R, S). outputs R=[5, 5, 5], S=9.
?- selectdiscard([5, 5, 5], [2, 5, 3], R, S). outputs R=[5, 5, 5], S=5.
?- selectdiscard([1, 4, 5], [2, 5], R, S). outputs R=[2, 5], S=10.
```

Exercises from old exams:



```
selectdiscard([], [], [], 0):- !.
selectdiscard([], [E|L], R, S):- selectdiscard([], L, R, S2), S is S2+E, !.
selectdiscard([E|L], [], R, S):- selectdiscard(L, [], R, S2), S is S2+E, !.
selectdiscard([E1|L1], [E2|L2], [GE|R], S):- E1>E2, selectdiscard(L1, L2, R, S2),
        GE is E1, S is S2+E2, !.
selectdiscard([E1|L1], [E2|L2], [GE|R], S):- E1==E2, selectdiscard(L1, L2, R, S),
        GE is E1, !.
selectdiscard([E1|L1], [E2|L2], [GE|R], S):- E1<E2, selectdiscard(L1, L2, R, S2),
        GE is E2, S is S2+E1.
```

Exercises from old exams:

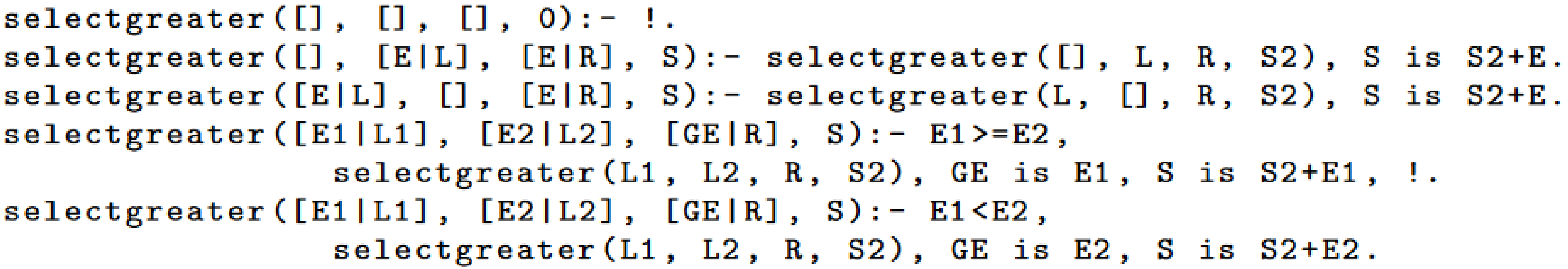


(5 points) Write a Prolog program that defines a predicate **selectgreater(L1, L2, R, S)**, that given 2 lists L1 and L2 compares their elements pairwise and returns the list R of the greater elements, along with the sum S of all the element in the list R. If one of the two lists has more elements than the other, the elements in such a list must be included in R.

Examples:

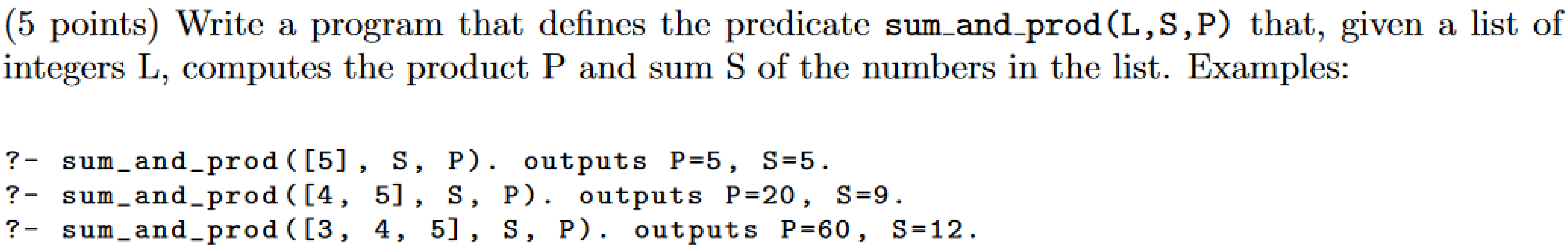
```
?- selectgreater([1, 4, 5], [2, 5, 3], R, S). outputs R=[2, 5, 5], S=12.
?- selectgreater([5, 4, 5], [2, 5, 3], R, S). outputs R=[5, 5, 5], S=15.
?- selectgreater([4, 5], [2, 5, 3], R, S). outputs R=[4, 5, 3], S=12.
```

Exercises from old exams:



```
selectgreater([], [], [], 0):- !.
selectgreater([], [E|L], [E|R], S):- selectgreater([], L, R, S2), S is S2+E.
selectgreater([E|L], [], [E|R], S):- selectgreater(L, [], R, S2), S is S2+E.
selectgreater([E1|L1], [E2|L2], [GE|R], S):- E1>=E2,
                selectgreater(L1, L2, R, S2), GE is E1, S is S2+E1, !.
selectgreater([E1|L1], [E2|L2], [GE|R], S):- E1<E2,
                selectgreater(L1, L2, R, S2), GE is E2, S is S2+E2.
```

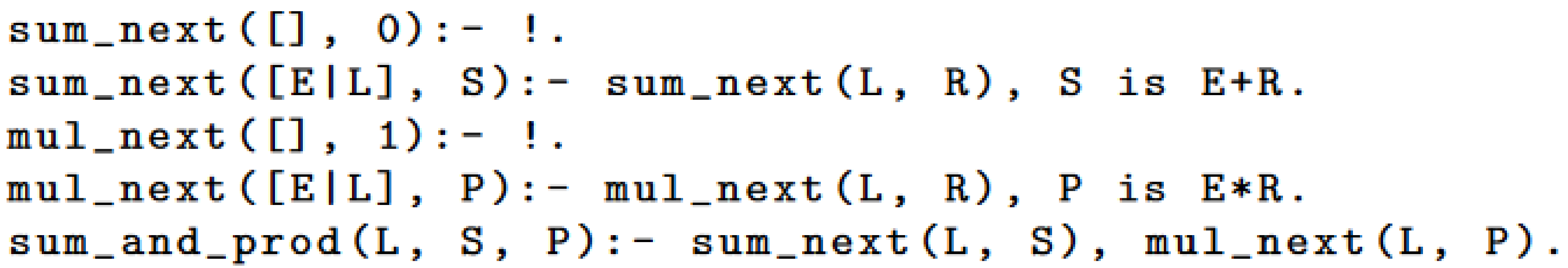
Exercises from old exams:



(5 points) Write a program that defines the predicate **sum_and_prod(L,S,P)** that, given a list of integers L, computes the product P and sum S of the numbers in the list. Examples:

```
?- sum_and_prod([5], S, P). outputs P=5, S=5.
?- sum_and_prod([4, 5], S, P). outputs P=20, S=9.
?- sum_and_prod([3, 4, 5], S, P). outputs P=60, S=12.
```

Exercises from old exams:



```
sum_next([], 0):- !.
sum_next([E|L], S):- sum_next(L, R), S is E+R.
mul_next([], 1):- !.
mul_next([E|L], P):- mul_next(L, R), P is E*R.
sum_and_prod(L, S, P):- sum_next(L, S), mul_next(L, P).
```

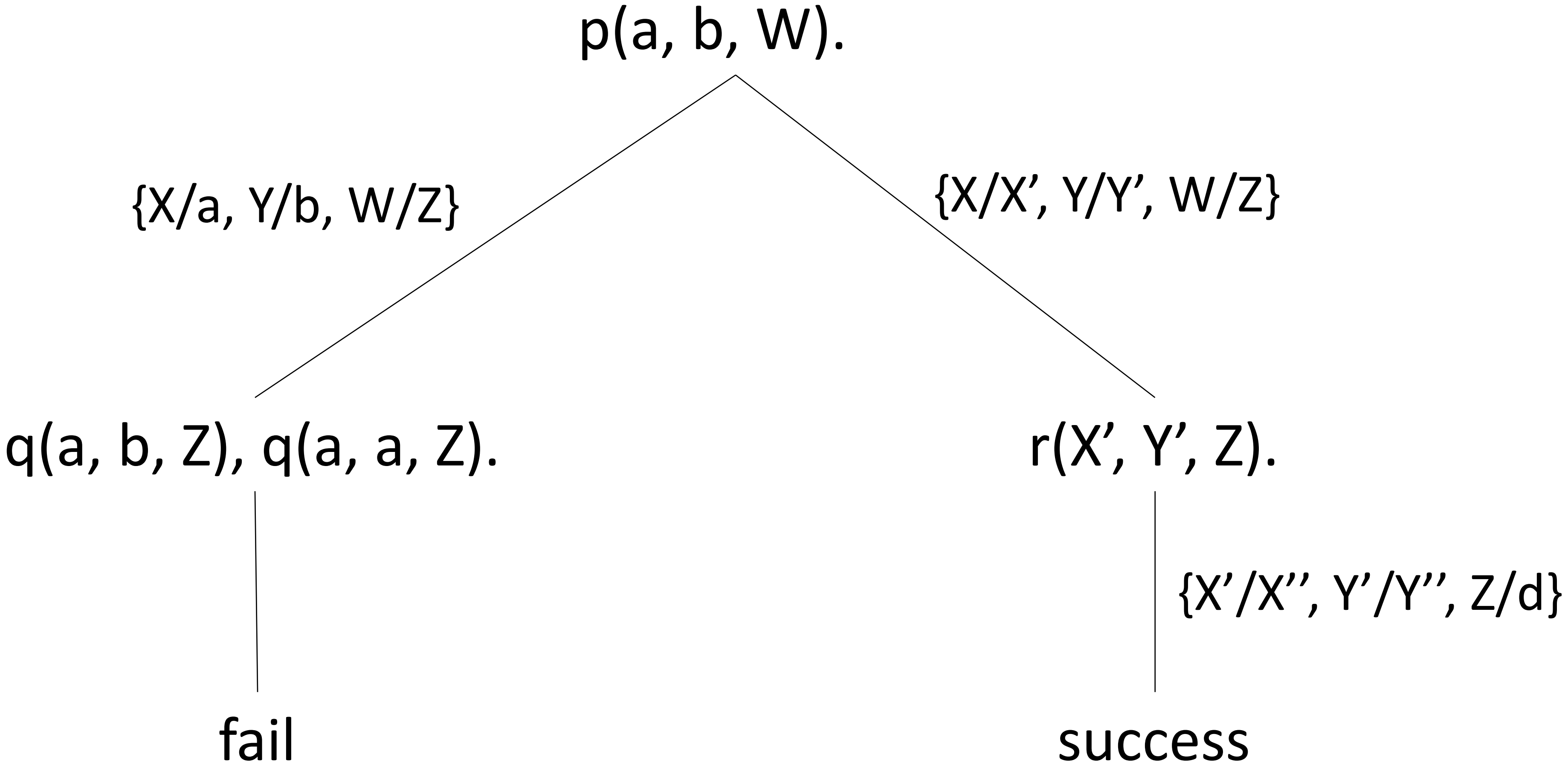
Prolog

Derivation

1) Show the derivation of the goal **p(a, b, W)** in the following Prolog program

```
p (X ,Y , Z ): - q (X ,Y , Z ) , q (X ,X , Z ).
p (X ,Y , Z ): - r (X ,Y , Z ).
q (X ,X , c ).
q (X ,a , c ).
r (X ,Y , d ).
```

1) Solution



p(a, b, W).

{X/a, Y/b, W/Z}

{X/X’, Y/Y’, W/Z}

q(a, b, Z), q(a, a, Z).

r(X’, Y’, Z).

{X’/X’’, Y’/Y’’, Z/d}

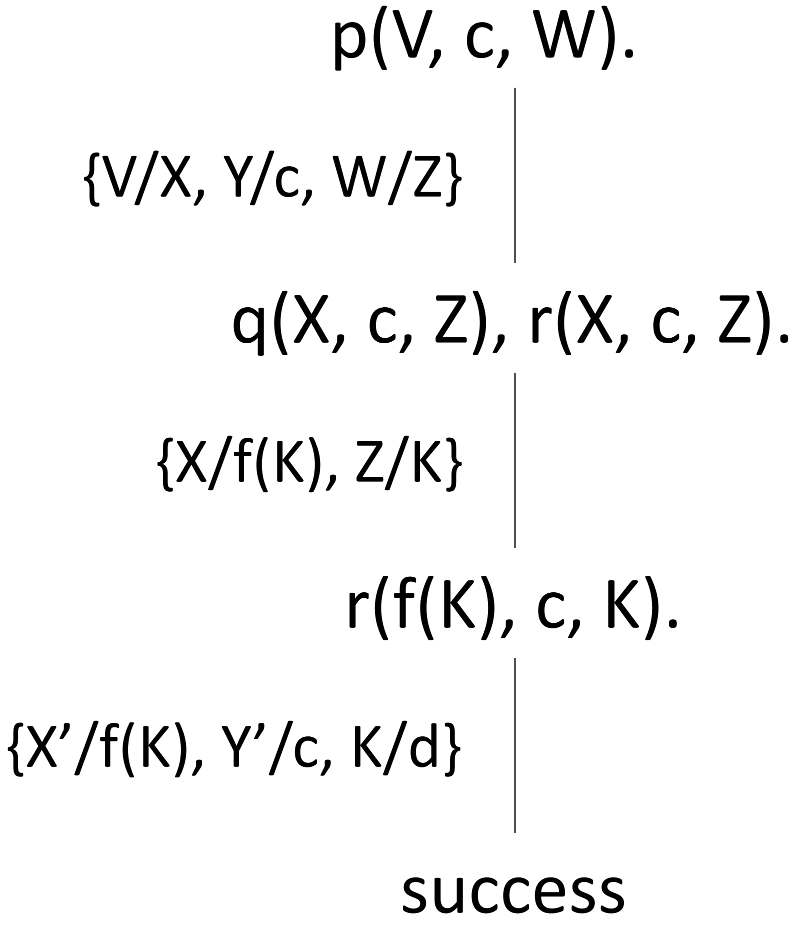
fail

success

2) Show the derivation of the goal **p(V, c, W)** in the following Prolog program

```
p (X ,b , Z ): - q (X ,Y , Z ) , q (X ,X , Z ).
p (X ,Y , Z ): - q (X ,Y , Z ) , r (X ,Y , Z ).
q (X ,b , c ).
q ( f ( K ) ,c , K ).
r (X ,Y , d ).
```

2) Solution



p(V, c, W).

{V/X, Y/c, W/Z}

q(X, c, Z), r(X, c, Z).

{X/f(K), Z/K}

r(f(K), c, K).

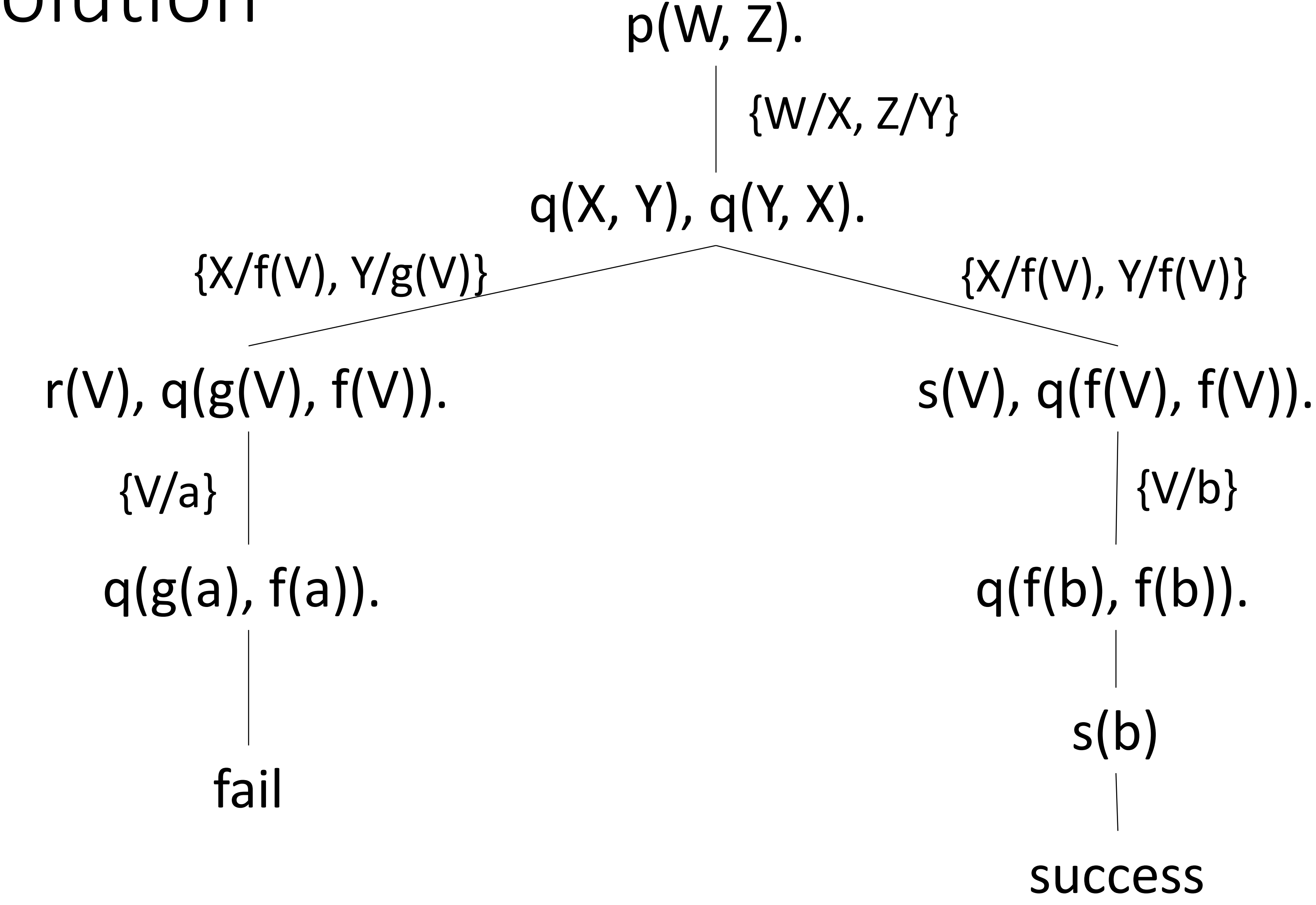
{X’/f(K), Y’/c, K/d}

success

3) Show the derivation of the goal **p(W, Z)** in the following Prolog program

```
p (X , Y ): - q (X , Y ) , q (Y , X ).
q ( f ( V ) , g ( V )): - r ( V ).
q ( f ( V ) , f ( V )): - s ( V ).
r ( a ).
s ( b ).
```

3) Solution



p(W, Z).

{W/X, Z/Y}

q(X, Y), q(Y, X).

{X/f(V), Y/g(V)}

{X/f(V), Y/f(V)}

r(V), q(g(V), f(V)).

s(V), q(f(V), f(V)).

{V/a}

{V/b}

q(g(a), f(a)).

q(f(b), f(b)).

s(b)

fail

success