```
1. Determine the outcome of the following Prolog queries (error, failure,
success)! In case of success, specify the resulting variable substitutions!
All queries are fed to the system independently.

(a) 4*2 = 8.
(b) [a,b] = [X|Y].
(c) U+V = 5+7+2.
(d) 2*3 is X*Y.
(e) A is 2*4, B = A+1.

Solution:

(a) failure
(b) X = a, Y = [b]
(c) U = 5+7, V = 2
(d) error
(e) A = 8, B = 8+1

2. Write down the canonical form or draw the tree form of the both left and
right hand sides of the following unifications. Specify the variable
substitutions which the unifications lead to.

(a) [X, a/X+b, Y+Z] = .(c, [U, U]).
(b) g([2*3|L], I*b) = g([W*V, V], a*J).

Solution:

(a) '.'(X,'.'(+(/(a,X),b),'.'(+(Y,Z),[])))
    '.'(c,'.'(U,'.'(U,[])))
    U = a/c+b, X = c, Y = a/c, Z = b

(b) g('.'(*(2,3),L),*(I,b))
    g('.'(*(W,V),'.'(V,[])),*(a,J))
    I = a, J = b, L = [3], V = 3, W = 2

3. Assume that the following program is loaded into the Prolog system.

q(X, 0) :- X >= 0, X =< 100.
q(X, 1) :- X > 100.

p([X|_], Y) :- q(X, Y).
p([_|L], Y) :- p(L, Y).

Determine the values
that X will take as a result of the following (independent) queries! Write
down all solutions separated by semicolons, in the same order as the system
would enumerate them! If there are no solutions, write {no}!

(a) p([], X).
(b) p([1], X).
(c) p([1000], X).
(d) p([1,1000,1], X).
(e) p([1,10,-10,100,-100,1000], X).
(f) Assume that in the call p(L, X) to the above predicate, the L argument
    is a list containing positive numbers. Describe in general what X
    values will be generated by the Prolog system and in what order.

Solution:

(a) {no}
(b) X = 0
(c) X = 1
(d) X = 0; X = 1; X = 0
(e) X = 0; X = 0; X = 0; X = 1
(f) For all elements of L which are between 0 and 100 (inclusive), X = 0 is
    returned, for all elements larger than 100, 1 is returned. Negative
    elements are ignored.  The order follows the order of elements in the
    list.
```



```
4. Consider a list consisting of X-Y pairs. Write a Prolog procedure which
counts those list elements for which the X*Y product is less than a given N
value. An efficient, tail recursive solution is appreciated, but not
required. If an auxiliary procedure is deemed necessary, write a
declarative head comment for it!

% smaller(+L, +N, -Cnt): There are Cnt number of elements in the list L
% consisting of X-Y pairs, for which X*Y is less than N. L and N are input
% parameters, Cnt is an output parameter.

Naive solution:

smaller([], _, 0).
smaller([X-Y|L], N, Cnt) :-
        smaller(L, N, Cnt0),
        (   X*Y < N
        ->  Cnt is Cnt0+1
        ;   Cnt = Cnt0
        ).

Tail recursive solution:

smaller(L, N, Cnt) :-
        smaller(L, N, 0, Cnt).

% smaller(L, N, Cnt0, Cnt): The number of X-Y elements in L for which X*Y <
% N is Cnt-Cnt0.
smaller([], _, Cnt, Cnt).
smaller([X-Y|L], N, Cnt0, Cnt) :-
        (   X*Y < N
        ->  Cnt1 is Cnt0+1
        ;   Cnt1 = Cnt0
        ),
        smaller(L, N, Cnt1, Cnt).

5. All of the following independent, syntactically correct declarations
have two semantic errors in them. Which are these?

(a) [op>(#"a", "b"), (1, 2) <> (1, 2, 3), true = false]
(b) (2*3 = 3+3, chr 95, ~9) = (6*1, "b", 0-5-4)
(c) foldl op@ [] [4, 2, 6, 4, 1, 2.0]

Solution:

(a) op>(#"a", "b"): character compared with string
    (1, 2) <> (1, 2, 3): comparing tuples of different size

(b) (2*3 = 3+3, ... = (6*1,...: boolean compared with integer
     ... chr 95 ... = ... "b" ...: character compared with string

(c) op@ requires two list operands, but the left operand is an integer
      taken from the list
    ...4, 1, 2.0]: list cannot mix types integer and real

6. What is the value of q after the evaluation of the following independent
declarations?

(a) val (_::_::_::_::q) = explode "eas" @ rev [#"r", #"e", #"t"]
(b) val (_::q::_) = List.map Char.isAlpha (explode "4r3e2a1d")
(c) val q = List.filter (fn (b, a) => a > b)
                [(7, 3*3), (1, 2), (ord #"Z", ord #"A")]

Solution:

(a) val q = [#"e", #"r"] : char list
(b) val q = true : bool
(c) val q = [(7, 9), (1, 2)] : (int * int) list
```

```
7. Consider the following function definitions!

fun zip (x::xs, y::ys) = (y, x) :: zip(xs, ys) | zip _ = []
fun f zs = zip(zs, tl zs)
fun g zs = map op− (f zs)

What is the value of x after the evaluation of the following independent
declarations?

(a) val x = g [~1]
(b) val x = g [~1,1]
(c) val x = g [1,3,6,10,15]
(d) List.filter op> (f [1,4,2,3,0])
(e) map op+ (List.filter op< (f [0,3,2,4,1]))

Solution:

(a) val x = [] : int list
(b) val x = [2] : int list
(c) val x = [2, 3, 4, 5] : int list
(d) val it = [(4, 1), (3, 2)] : (int * int) list
(e) val it = [5, 5] : int list


8. We call three neighboring elements of an integer list a sum triplet
resp. difference triplet, if the sum resp. difference of the first and the
third elements is equal to the middle. Write an SML function called
sumdiff, which returns true if and only if the list provided in its
argument contains a sum or a difference triplet. You may define auxilliary
functions if you write declarative head comments for them.

(* sumdiff : int list −> bool
   sumdiff zs = true iff zs contains sum or difference triplets *)

Solution:

fun sumdiff (x::(yzs as y::z::_)) =
    x+z = y orelse x−z = y orelse sumdiff yzs
  | sumdiff _ = false
```