```
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) Z=1+5,\+Z=2*3                          ----> Z=1+5
(b) X is 4−3,Y is X+1,Y=3−1                ----> failure
(c) D=3+E,\+D=2,R is D+1                   ----> error
(d) append([],[a,12|_],[A,_])              ----> A=a
(e) 2+4−2=A−B                              ----> A=2+4, B=2

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,[3*_]|Z]=[Y,[Y*2]]                  ----> X=3, Y=3, Z=[]
    left:  .(X,.(.(*(3,_),[]),Z))
    right: .(Y,.(.(*(Y,2),[]),[]))

(b) f(_+A*a,[C,_|B],F)=f(F+C,[3*_,b],6)    ----> A=3, B=[], C=3*a, F=6
    left:  f(+(_,*(A,a)),.(C,.(_,B)),F)
    right: f(+(F,C),.(*(3,_),.(b,[])),6)

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

p([A,B|_], T, E) :−
   A > T,
   A < B,
   E = A.
p([A|As], _, E):−
   p(As, A, E).

Determine the values that A 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([2,4,1.2,3],0,A)                     ----> A = 2
(b) p([1,5,10],2,A)                        ----> A = 5
(c) p([3,4,1,5,8],1,A)                     ----> A = 3 ; 5
(d) p([3,4,2,5,3,2],4,A)                   ----> {no}
(e) p([2,4,6,7,8,2,4,5],3,A)               ----> A = 4 ; 6 ; 7 ; 4

Consider the following procedure, which uses the \texttt{p/3} predicate
defined above:

% p(L, Z): Z is a member of the L list such that...
p(L, Z) :− L = [A|As], p(As, A, Z).

(f) Describe in a declarative manner what this p/2 predicate does by
    completing the above head comment. Make sure to specify the enumeration
    order of the solutions

----> Z is a member of the L list such that Z is larger than its
      predecessor but smaller than its successor in the list.  The elements
      are returned in the same order as they appear in the list.

4. Consider a list consisting of X−Y pairs.  We call the pairs A−B and C−D
mergeable, if either A=C or B=D holds, and they can be merged into the pair
X−Y, where X=A+C and Y=B+D.  Write a Prolog procedure called merged which,
given a list of such pairs as input in its first argument, enumerates the
merged value of all mergeable neighboring list elements, preserving the
order in which they appear in the list.  Enumerate each merged value
exactly once!  If an auxiliary procedure is deemed necessary, write a
declarative head comment for it!
```



```
% merged(XYs, M): M is a merged value of two neighboring pairs in XYs.
merged([A−B,C−D|_], X−Y) :−
       (   A =:= C
       −>  true
       ;   B =:= D
       ),
       X is A+C,
       Y is B+D.
merged([_|T], XY) :−
       merged(T, XY).

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

(a) (#"a"::#"b" = explode "ab", (1, 2) < (2, 1), 1 < 2 < 3)

----> Actually, there are three errors here:
      1) #"a"::[#"b"]   : op:: expects a list as its second argument
      2) (1, 2) < (2, 1): pairs (tuples in general) cannot be compared
      3) 1 < 2 < 3:     : boolean (1 < 2) cannot be compared with int (3)

(b) [3+3, chr 93.0, 7] = [3*2, ord #"b", 0−3−4, 0]

----> 1) chr expects an int argument, 93.0 is real
      2) all elements of a list must be of the same type, but chr returns char

(c) map (op +) [65, 6+5, ord chr 65]

----> 1) op+ expects pairs as arguments, this list contains integers
      2) ord (chr 65): without parentheses, ord would get two arguments:
         chr and 65.

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

(a) val (_::_::_::x) = rev(explode "PL" @ [#"S", #"M", #"L"])
----> x = [#"L", #"P"]

(b) val (_::x::_) = List.filter (not o Char.isUpper) (explode "aBcDeF")
----> x = #"c"

(c) val x = #1(foldr (fn (x, (y, b)) => (x+y, b andalso x < y))
                     (0, true)
                     [3,2,1])
----> x = 6

7. Consider the following function definitions!

(* val f1 = fn : string list * string list −> string list −> string list
   val f2 = fn : string list * string list −> string list *)
fun f1 (m::ms, n::ns) rs = f1 (ms, ns) (m^n::rs)
  | f1 _ rs = rs
and f2 msns = f1 msns []

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

(a) Show the evaluation steps of f2 (["SM","Pro"],["L","log"]) using the
substitutuon model and eager evaluation!

f2 (["SM","Pro"],["L","log"]) −−> f1 (["SM","Pro"],["L","log"]) [] −−>
−−> f1 (["Pro"], ["log"]) ["SML"] −−> f1 ([], []) ["Prolog", "SML"] −−>
−−> ["Prolog", "SML"]

(1) x = f2 ([],[])                                  ----> x = []
(2) x = f2 (["a","b"],["c"])                        ----> x = ["ac"]
(3) x = f1 (["L"],["I","J"]) ["SP"]                 ----> x = ["LI", "SP"]
(4) x = f1 (["Er"],["la","nguage"]) (f2 (["n"],["g"]))
                                                    ----> x = ["Erla", "ng"]
```

```
8. A word (char list) is called a TLA (Three Letter Abbreviation), if it
consists of 3 capital letters. Write an SML function called firstTLA which
returns the first TLA found in its argument of type char list list, and
throws a notfound exception, if the list doesn't contain TLA's. You may
define auxiliary functions only with appropriate head−comment!

(* firstTLA : char list list −> char list
   firstTLA l = the first TLA found in l *)
*)

fun firstTLA css =
    let fun isTLA [a,b,c] = List.all Char.isUpper [a,b,c]
          | isTLA _ = false
    in
        case List.find isTLA css of
            SOME tla => tla
          | NONE => raise notfound
    end
```