22. For a heterogeneous list, formed from positive integer numbers and lists of positive integer numbers, write a predicate to create a list that includes the positions where the minimum element is in the list (if the minimum is within a sub-list, the position of the sub-list will be added).

a) [] → []
b) [1, [2, 3], 10, [4, 4], 3, 6, [7, 10, 1], 5, [1, 1, 1], 7] → [1, 7, 9]
c) [1, 2, 3, 10, 4, 5, 6, 10, 1] → [1, 9]

```prolog
% Student exercise profile
:- set_prolog_flag(occurs_check, error).      % disallow cyclic terms
:- set_prolog_stack(global, limit(8 000 000)).  % limit term space (8Mb)
:- set_prolog_stack(local,  limit(2 000 000)).  % limit environment space

% Your program goes here

min(A, B, A) :- A =< B, !.
min(A, B, B) :- A > B.

min_hetero([X], X) :-
    \+ is_list(X).

min_hetero([H|T], Min) :-
    \+ is_list(H),
    min_hetero(T, MinT),
    min(H, MinT, Min).

min_hetero([H|T], Min) :-
    is_list(H),
    min_hetero(H, MinH),
    min_hetero(T, MinT),
    min(MinH, MinT, Min).

sublist_contains_min([X|_], Min) :-
    \+ is_list(X),
    X = Min, !.

sublist_contains_min([H|T], Min) :-
    \+ is_list(H),
    H \= Min,
    sublist_contains_min(T, Min).

sublist_contains_min([H|_], Min) :-
    is_list(H),
    sublist_contains_min(H, Min), !.

min_positions([], _, _, []).

min_positions([H|T], Min, I, [I|Rest]) :-
    \+ is_list(H),
    H = Min, !,
    I1 is I + 1,
```

```prolog
    min_positions(T, Min, I1, Rest).

min_positions([H|T], Min, I, [I|Rest]) :-
    is_list(H),
    H \= [],
    sublist_contains_min(H, Min), !,
    I1 is I + 1,
    min_positions(T, Min, I1, Rest).

min_positions([_|T], Min, I, Rest) :-
    I1 is I + 1,
    min_positions(T, Min, I1, Rest).

min_pos_list([], []).
min_pos_list(L, PosList) :-
    L \= [],
    min_hetero(L, Min),
    min_positions(L, Min, 1, PosList).



/** <examples> Your example queries go here, e.g.
?- min_pos_list([], P). -> []
min_pos_list([1,2,3,10,4,5,6,10,1], P). -> [1,9]
min_pos_list([1,[2,3],10,[4,4],3,6,[7,10,1],5,[1,1,1],7], P). -> [1,7,9]

*/
```

```
min_pos_list([], P).
```

P = []

Next | 10 | 100 | 1,000 | Stop

?-
```
min_pos_list([], P).
```

```
min_pos_list([1,2,3,10,4,5,6,10,1], P).
```

P = [1, 9]

Next | 10 | 100 | 1,000 | Stop

?-
```
min_pos_list([1,2,3,10,4,5,6,10,1], P).
```

```
min_pos_list([1,[2,3],10,[4,4],3,6,[7,10,1],5,[1,1,1],7], P).
```

P = [1, 7, 9]

Next | 10 | 100 | 1,000 | Stop

?-
```
min_pos_list([1,[2,3],10,[4,4],3,6,[7,10,1],5,[1,1,1],7], P).
```

*Mathematical Model*
> \+   -> input
> \-   -> output

$min(A, B, Min) = A$, if $A \leq B$
$= B$, if $A > B$
*Flow: min(+A, +B, -Min)*

*Flow: min_hetero(+L, -Min)*

*(1) min_hetero([x], x) $\iff$ x $\notin$ List (x is not a list, x is an integer)*
*(2) min_hetero([h | T], Min) where h $\notin$ List $\iff$ min_hetero(T, MinT) $\land$ min(h, MinT, Min)*

*(3) min_hetero([h | T], Min) where h ∈ List ⇔ min_hetero(h, MinH) ∧ min_hetero(T, MinT) ∧ min(MinH, MinT, Min)*

*Flow: sublist_contains_min(+L, +Min)*
*(1) sublist_contains_min([x | _], Min) ⇔ x ∉ List ∧ x = Min*
*(2) sublist_contains_min([h | T], Min) where h ∉ List ⇔ h ≠ Min ∧ sublist_contains_min(T, Min)*
*(3) sublist_contains_min([h | _], Min) where h ∈ List ⇔ sublist_contains_min(h, Min)*

*Flow: min_positions(+L, +Min, +I, -Positions)*
*(1) min_positions([], Min, I, [])*
*(2) min_positions([h | T], Min, I, [I | Rest]) where h ∉ List ⇔ h = Min ∧ min_positions(T, Min, I+1, Rest)*
*(3) min_positions([h | T], Min, I, [I | Rest]) where h ∈ List ⇔ % h ≠ [] ∧ sublist_contains_min(h, Min) ∧ min_positions(T, Min, I+1, Rest)*
*(4) min_positions([h | T], Min, I, Rest) ⇔ (h is integer ∧ h ≠ Min) ∨ (h is list ∧ sublist_contains_min(h, Min)) ∧ min_positions(T, Min, I+1, Rest)*


*Flow: min_pos_list(+L, -PosList)*
*(1) min_pos_list([], [])*
*(2) min_pos_list(L, PosList) where L ≠ [] ⇔ min_hetero(L, Min) ∧ min_positions(L, Min, 1, PosList)*