10 Prolog (ijl20)

When answering this question ensure each relation has a comment giving a declarative reading of its behaviour. You should avoid unnecessary use of cut and not use extra-logical relations such as `findall`, `assertz` and `not` (`\+`). Built-in library relations should not be assumed. The `notmember` relation given in the first part may be re-used if required.

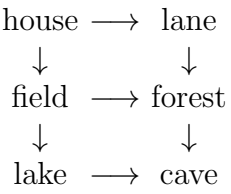
(*a*) Assume the built-in operator `\=` meaning *not unifiable with*, and a relation `notmember(+A,+L)` defined thus:

```
notmember(_,[]).
notmember(A,[H|T]) :- A \= H, notmember(A,T).
```

Explain where *facts*, *rules*, *atoms*, *compound terms* have been used. Why does `notmember(A,[a,b,c])` fail? [2 marks]

(*b*) Write a `reverse(+A,?B)` relation suitable for *last call optimisation*. What makes it suitable for LCO? [3 marks]

(*c*) This small diagram represents our world map, with the arrows representing downhill lanes between places on the map.



Represent these downhill lanes with a `downhill(?A,?B)` relation. [2 marks]

(*d*) Assuming `downhill(?A,?B)` is acyclic, define a relation `downhill_path(?A,?B)` which succeeds if place `B` can be reached from place `A` along downhill lanes. [2 marks]

(*e*) Define a relation `linked(?A,?B)` which succeeds if a lane *directly* connects places `A` and `B` downhill or the reverse, e.g. `:- linked(cave,forest)` should succeed. [2 marks]

(*f*) Define a relation `linked_path(+A,+B,?Path)` which finds a linked path between places `A` and `B`, reporting the ordered list of places visited from `A` to `B` in the `Path` argument. [6 marks]

(*g*) Assume a relation `danger(?A,?D)` where `D` gives a numerical value for the danger at each place `A`, for example `:- danger(forest,X)` might succeed with `X=4`. Extend your `linked_path` relation so that it also returns the sum of the danger values along the path, i.e. `linked_path(+A,+B,?Path,?Danger)` [3 marks]