7 Prolog (ACR)

In this question you should ensure that your predicates behave appropriately with backtracking and avoid over-use of cut. You should provide an implementation of any library predicates used. You may not make use of extra-logical built-in predicates such as findAll. Minor syntactic errors will not be penalised.

(a) Explain the operation of cut (!) in a Prolog program. [2 marks]

(b) Rewrite choose without using cut. [2 marks]

choose(0,_,[]) :- !.
choose(N,[H|T],[H|R]) :- M is N-1, choose(M,T,R).
choose(N,[_|T],R) :- choose(N,T,R).

(c) Explain the operation of not (also written as \+) in a Prolog program. [1 mark]

(d) Rewrite chooseAll without using not and cut (!). [10 marks]

chooseAll(N,L,Res) :- chooseAll(N,L,[],Res).
chooseAll(N,L,Seen,Res) :- choose(N,L,R),
    not(member(R,Seen)), !,
    chooseAll(N,L,[R|Seen],Res).
chooseAll(\_,\_,Res,Res).

(e) What is Last Call Optimisation and why is it beneficial? [3 marks]

(f) Rewrite pos to enable Last Call Optimisation. [2 marks]

pos([],[]).
pos([H|T],[H|R]) :- H >= 0, pos(T,R).
pos([_|T],R) :- H < 0, pos(T,R).