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]

\[
\begin{align*}
\text{choose}(0,&\_,[\_]) :- !. \\
\text{choose}(N,[H|T],[H|R]) :- M \text{ is } N-1, \text{ choose}(M,T,R). \\
\text{choose}(N,[\_|T],R) :- \text{ choose}(N,T,R).
\end{align*}
\]

(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]

\[
\begin{align*}
\text{chooseAll}(N,L,\_,Res) &:- \text{ chooseAll}(N,L,[\_],Res). \\
\text{chooseAll}(N,L,Seen,Res) &:- \text{ choose}(N,L,R), \\
&\quad \text{ not(member}(R,\text{ Seen})), !, \\
&\quad \text{ chooseAll}(N,L,[R|\text{ Seen}],Res). \\
\text{chooseAll}(\_,\_,\_,Res,Res).
\end{align*}
\]

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

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

\[
\begin{align*}
pos([\_],[\_]). \\
pos([H|T],[H|R]) &:- H \geq 0, \text{ pos}(T,R). \\
pos([H|T],R) &:- H < 0, \text{ pos}(T,R).
\end{align*}
\]