The Prolog predicate \texttt{perm(+In,-Out)} generates all permutations of the input list \texttt{In}. A programmer implements \texttt{perm/2} as follows:

\begin{verbatim}
perm([],[]).
perm([H|T],H,T) :- take([H|T],H,R), perm(R,T).
\end{verbatim}

The predicate \texttt{take(+L,-E,-R)} removes one element (E) from the input list \texttt{L} and unifies \texttt{R} with the remainder of \texttt{L}. Thus, the list \texttt{R} has one element fewer than \texttt{L}.

\textbf{(a)} Consider the \texttt{perm/2} predicate:

\begin{itemize}
  \item[(i)] Explain briefly in words the operation of the \texttt{perm/2} predicate. \hspace{1cm} [3 marks]
  \item[(ii)] Provide an implementation of the \texttt{take/3} predicate. \hspace{1cm} [4 marks]
  \item[(iii)] Give the complete sequence of answers (in the correct order) generated by \texttt{perm([1,2,3],A)}. \hspace{1cm} [3 marks]
\end{itemize}

\textbf{(b)} A student attempts to invoke the query \texttt{perm(A,[1,2,3])}.

\begin{itemize}
  \item[(i)] Explain what happens and why. \hspace{1cm} [5 marks]
  \item[(ii)] Implement a predicate \texttt{sameLength/2} which is true if the two parameters are lists of the same length. \hspace{1cm} [2 marks]
  \item[(iii)] Using \texttt{sameLength/2}, or otherwise, provide an implementation of \texttt{safePerm/2} which generates permutations regardless of the order in which the parameters are provided: both \texttt{safePerm(+In,-Out)} and \texttt{safePerm(-Out,+In)} should generate all permutations of \texttt{In}. The order in which these permutations are generated is not important. \hspace{1cm} [3 marks]
\end{itemize}