Question 9A (10 points): Consider the following two similar problems.

\[ \text{LONG}(G, v, w, k) \]: Is there a simple path of \textbf{length at least} \( k \) from node \( v \) to node \( w \) in directed graph \( G \)?

\[ \text{LONGEST}(G, v, w, k) \]: Does the \textbf{longest} simple path from node \( v \) to node \( w \) in directed graph \( G \) have length \( k \)?

In both cases, assume length is just the number of edges in the path.

Suppose \( A \) is a polynomial-time algorithm for the \text{LONG} problem. Explain why there then must be a polynomial-time algorithm \( B \) for the \text{LONGEST} problem. Assume for both \( A \) and \( B \) that the graph \( G \) is given as a list of nodes plus a list of edges.

\textit{Given algorithm} \( A \), \textit{we can construct} \( B \) \textit{using two calls to} \( A \):

\[
B\text{-Longest}(G, v, w, k) \\
\text{if } A\text{-Long}(G, v, w, k) \text{ and not } A\text{-Long}(G, v, w, k+1) \text{ then return true} \\
\text{else return false}
\]

The longest \( v \)–\( w \) path in \( G \) has length \( k \) if there is a path of length at least \( k \), but no path of length \( k+1 \) or longer. The conversion of input to \( B \) to inputs to \( A \) is trivial: \( O(1) \). Thus we have a polynomial-time reduction to two calls of a polynomial-time algorithm, so \( B \) runs in polynomial time.