Problem 1. Give a deterministic algorithm for the language

\[
LP = \{\langle G, s, t, k \rangle \mid \text{There is a path in } G \text{ from } s \text{ to } t \text{ of length } k \},
\]

where \( G = (V, E) \) is an unweighted graph, \( s \) and \( t \) belong to \( V \), and \( k \in \mathbb{N} \). Here, the length of a path is the number of edges you cross over from its beginning to its end.

Problem 2. Nondeterministic Turing machines are best used for sufficiently complex languages that we can benefit from exploring multiple paths. A consequence of this is that implementing a nondeterministic Turing machine at its lowest level via the formal definition is somewhat of a nightmare. As such, we usually only give psuedocode algorithms for them.

Give a nondeterministic Turing machine (you need not draw one) which decides \( LP \) at least exponentially faster than the deterministic version.

Problem 3. Explain in your own words why it is hard to flip the output of a nondeterministic Turing machine.