Problems

1. **Experiment with the nonsense word generator with different prefix sizes as per the code in the `lec15` directory of the code repo**. Can you think of a way of deciding what size of prefix generates more word-like words?
2. One problem with the nonsense word generator is that it tends to produce rather longer words than normal since it has no inbuilt mechanism to favour ending a word once a certain length is reached. **Suggest one or more approaches that would favour ending words more frequently as the length increased**. You don't need to write code to implement your suggestions.
3. Implement the naive recursive and memoised edit distance algorithms discussed in Lecture 22 (`RecursiveLCS` and `MemoLCSRecurse`). Some details/alterations you'll need to make: return a distance instead of a string, and when computing the minimum, there will be 3 recursive calls instead of two.
4. **Extension Exercise**: Use the edit distance code to measure whether nonsense words with prefix histories of size 1, 2, 3, 4 or 5 are the most English-like. You could generate, say, 100 nonsense words with a prefix of 1, then measure the edit distance from those words to the closest word in the dictionary and average over the 100 words. Then repeat for prefixes of size 2 or 3 etc. Which prefix length has the shortest average distance, and is this a good measure?