This solution converts phone numbers into possible word combinations using a given dictionary and a digit-to-character mapping (like on a classic phone keypad).

Main Components

- Dictionary Preprocessing: Each word from the dictionary is mapped into its numeric representation using the digitToChars mapping (e.g., HELLO → 43556). These are stored in a Map for quick lookup.
- Input Normalization:Phone numbers are read from a file or STDIN and cleaned of punctuation/spaces.

Core Algorithm: function phonewordDP -> Dynamic Programming with Memoization

The phonewordDP function uses a recursive dynamic programming approach to find all possible word combinations that match a given phone number.

It defines a function **dp(index, usedNumber)** that attempts to decode the phone number starting from the given **index**. The parameter **usedNumber** is a boolean flag indicating whether a digit has already been left unchanged in the current path, so no two consecutive digits remain unchanged.

The logic of the recursion:

1. When index reaches the end of the phone number, it returns a list containing an empty sequence, indicating a valid match path.
2. Starting from index, it attempts to extract substrings of the phone number with lengths between the minimum and maximum dictionary word lengths.
3. For each substring that matches a dictionary word (by checking the Map), it recursively calls dp from the position after that substring with usedNumber reset to false (because no digit was left as-is here).
4. If no digit has been left unchanged yet (usedNumber is false), it tries leaving the current digit as-is (not converted), marks usedNumber as true, and continues recursively from the next index.
5. Results of each (index, usedNumber) state are memoized to avoid redundant computations.
6. Finally, all valid sequences of words and digits are combined and returned as strings joined by dashes (-).