# Section 1 The essence of your solution to the challenge in a single paragraph.

In my design, I apply an external database and use an algorithm on the input values to fuse the resulting words. Finally, the words are output in order from high to low by calculating the frequency and relevance scores of the word array.

# Section 2 An explanation of the original algorithms of your solution in non-technical language. You do not need to describe the workings of any standard algorithm, such as mergesort, that might comprise part of your solution.

This article mainly includes three algorithms – Phonetic Matching Algorithms, Word End Matching Algorithms, and Rank Algorithm (Original).

Through Phonetic Matching Algorithms, I convert the user input value into an output value containing three syllables, as the corresponding symbol of each word, and then compare it with the conversion database to get Words with the same symbol as the input word.

With Word End Matching Algorithms, I'm looking for words that end in the same ending as the user input value because that makes it rhyme.

Through Rank Algorithm, I score the frequency and relevance of the preliminary output word array, and output the words in order from high to low scores.

# Section 3 Pseudocode for each original algorithm. You are urged to follow the pseudocode conventions of Cormen at al, Chapter 2. Pay great attention to how you lay out your pseudocode. In particular, take care to use structured indentation. Pseudocode that is not correctly indented, or is otherwise unreadable, will not be marked.

# Section 4 A list of the data structures in your solution. Explain why each chosen data structure is suitable for the task.

# Section 5 A javascript implementation of your solution and a link to a one minute video that shows your code in execution. The source code must be visible and your code must be commented in order to facilitate cross-checking with your pseudocode.

# Section 6 Point out any defects of your design and/or implementation. Suggest remedies for these shortcomings.

To build an algorithm for suggesting rhymes, the steps including:

**Rhyme Database:**

Create a database of rhyming words. You can utilize existing resources or compile your own list of words grouped by their rhyming patterns.

**Phonetic Matching:**

Implement a phonetic matching algorithm to identify words with similar sounds. This could involve using algorithms like Soundex or Metaphone to compare the phonetic representations of words.

**Word End Matching:**

Compare the last syllables or phonemes of the submission word with those in the database. This helps identify words that end similarly, contributing to rhyme suggestions.

**Frequency and Relevance:**

Prioritize rhyming words based on frequency and relevance. Words that are commonly used in poetry or have a closer semantic connection to the submission word can be given higher priority.

**User Feedback Integration:**

Incorporate a mechanism for user feedback to enhance the algorithm's learning over time. This ensures the system adapts to individual preferences and writing styles.

**Continuous Improvement:**

Regularly update the rhyme database and algorithm to include new words and improve rhyme suggestions based on evolving language usage.

# Soundex Algorithm:

The Soundex algorithm is a phonetic algorithm designed to encode words based on their pronunciation, particularly in English. Its primary purpose is to convert words into a phonetic core, allowing for comparison of words that sound similar but may be spelled differently. Here's a brief overview of how the Soundex algorithm works:

**Word Breakdown:**

Soundex breaks down words into their phonetic components, focusing on the initial consonant and ignoring vowels and consecutive duplicate consonants.

**Encoding Scheme:**

Words are encoded into a four-character code. The first character represents the initial letter of the word, while the remaining three characters denote specific consonant sounds.

**Numeric Representation:**

Each encoded character corresponds to a numeric value. This numeric representation facilitates the comparison of words with similar sounds.

**Homophones Matching:**

The goal of Soundex is to encode homophones to the same representation, enabling efficient retrieval of words with similar pronunciations.

**Usage:**

Soundex is often used in databases and information retrieval systems for phonetic searching, assisting in finding words that have similar sounds, even if spelled differently.