# Documentation: Understanding the Word Pattern Problem

The **Word Pattern Problem** involves determining whether a string s follows a given pattern pattern. The concept of "following the pattern" requires a one-to-one mapping between the characters in the pattern and the words in s. This relationship ensures that each character in the pattern uniquely corresponds to a single word in s, and each word in s uniquely corresponds to a single character in the pattern. The problem becomes invalid if any character maps to multiple words or if any word maps to various characters.

The solution begins by splitting the string s into a list of words and verifying that the number of words matches the length of the pattern. If they differ, s cannot follow the pattern pattern. After verifying this, the algorithm uses two hash maps (dictionaries): one to map characters from the pattern pattern to words in s and another to map words in s back to characters in the pattern pattern. This dual-mapping approach ensures the bijection between the two elements is maintained.

As the algorithm processes each character-word pair from pattern and s, it checks whether the character already maps to a word. If a mismatch is found, the pattern is broken, and the function returns false. Similarly, it verifies that each word in s maps correctly to the corresponding character. Any inconsistency in either mapping results in an immediate failure to match the pattern. If all pairs are processed without conflicts, the function concludes that s follows the pattern.

This problem has several real-world parallels, such as verifying if a specific rule or code maps to a unique outcome or interpreting symbolic data to match predefined patterns. The constraints are designed to ensure clarity in input formatting, such as non-empty strings, no leading or trailing spaces, and words separated by single spaces. These constraints simplify implementation and reduce ambiguity in interpretation.

Overall, the Word Pattern Problem is a clasic example of using hash maps to solve a problem requiring a bijective relationship. It demonstrates fundamental programming concepts such as mapping, validation, and error handling, while also showcasing practical applications in validating structured input against predefined rules. The efficient use of data structures ensures the solution operates within acceptable time and space complexity limits.