Design Add and Search Words Data Structure

Core Idea:

- Use a tree.
- addWord: standard tree insert.
- search: handle '.' via DFS backtracking:
  - If current pattern char is a letter → follow that one child.
  - If it's '.' → try all non-null children from this node.
  - Success if you reach the end of the pattern at a node with isEnd = true.

This works well because constraints are small (≤25 length, ≤2 dots), so branching is limited.

Node Design:

- bool isEnd
- TrieNode* next[26] (or unordered_map; array is fine here).

Complexity:

Let L = word length, K = branching from a dot (≤26, but in practice smaller).

- addWord: O(L)
- search: no dots O(L)
  - with dots: worst-case O(K^d * L), where d ≤ 2 (so still small)