**ReadMe**

**A Quick Look**

Hey there! This C++ program is designed to find the longest and second-longest "compounded" words from a list of words. Now, you might be wondering what a "compounded" word is. Well, it's simply a word that's made up of two or more words from the list.

**How It Works**

The program is pretty straightforward and efficient. It uses an unordered set (think of it as a super-fast dictionary) to store the words for quick lookup. Then, it uses a recursive function (a function that calls itself) to check if a word can be formed by combining other words in the list. The program goes through each word in the list, checks if it's compounded, and keeps track of the longest and second-longest compounded words it finds.

**How to Run It**

1. Save the C++ code in a file, let's call it compound\_word.cpp.

2. Compile the C++ code using a compiler (like g++). You can use this command: g++ compound\_word.cpp -o compound\_word

3. Make sure that the input file "Input\_02.txt" is in the same directory as the compiled executable.

4. Run the program using this command: ./compound\_word

**Design Decisions & Approach**

I decided to use an unordered set for storing the words to ensure quick lookup times when checking if a word exists in the list. This data structure provides constant time average complexity for search, insert, and delete operations which makes it an ideal choice for this problem.

The recursive function isCompounded was used to check if a word can be formed by combining other words in the list. This approach simplifies the problem by breaking it down into smaller sub-problems.

The program reads words from an input file into a list, calls findLongestCompoundedWords to find the longest and second-longest compounded words, and then prints these words along with the time taken to process the file. This approach ensures that all words are considered and the longest compounded words are found efficiently.