

# Problem 843: Guess the Word

## Problem Information

**Difficulty:** Hard

**Acceptance Rate:** 37.16%

**Paid Only:** No

**Tags:** Array, Math, String, Interactive, Game Theory

## Problem Description

You are given an array of unique strings `words` where `words[i]` is six letters long. One word of `words` was chosen as a secret word.

You are also given the helper object `Master`. You may call `Master.guess(word)` where `word` is a six-letter-long string, and it must be from `words`. `Master.guess(word)` returns:

\* ``-1`` if `word` is not from `words`, or \* an integer representing the number of exact matches (value and position) of your guess to the secret word.

There is a parameter `allowedGuesses` for each test case where `allowedGuesses` is the maximum number of times you can call `Master.guess(word)`.

For each test case, you should call `Master.guess` with the secret word without exceeding the maximum number of allowed guesses. You will get:

\* \*\*\*"Either you took too many guesses, or you did not find the secret word."\*\* if you called `Master.guess` more than `allowedGuesses` times or if you did not call `Master.guess` with the secret word, or \* \*\*\*"You guessed the secret word correctly."\*\* if you called `Master.guess` with the secret word with the number of calls to `Master.guess` less than or equal to `allowedGuesses`.

The test cases are generated such that you can guess the secret word with a reasonable strategy (other than using the bruteforce method).

\*\*Example 1:\*\*

**\*\*Input:\*\*** secret = "acckzz", words = ["acckzz", "ccbazz", "eiowzz", "abcczz"], allowedGuesses = 10  
**\*\*Output:\*\*** You guessed the secret word correctly.  
**\*\*Explanation:\*\***  
master.guess("aaaaaa") returns -1, because "aaaaaa" is not in wordlist.  
master.guess("acckzz") returns 6, because "acckzz" is secret and has all 6 matches.  
master.guess("ccbazz") returns 3, because "ccbazz" has 3 matches. master.guess("eiowzz") returns 2, because "eiowzz" has 2 matches. master.guess("abcczz") returns 4, because "abcczz" has 4 matches. We made 5 calls to master.guess, and one of them was the secret, so we pass the test case.

**\*\*Example 2:\*\***

**\*\*Input:\*\*** secret = "hamada", words = ["hamada", "khaled"], allowedGuesses = 10  
**\*\*Output:\*\*** You guessed the secret word correctly.  
**\*\*Explanation:\*\*** Since there are two words, you can guess both.

**\*\*Constraints:\*\***

\* `1 <= words.length <= 100` \* `words[i].length == 6` \* `words[i]` consist of lowercase English letters.  
\* All the strings of `wordlist` are **unique**.  
\* `secret` exists in `words`. \* `10 <= allowedGuesses <= 30`

## Code Snippets

**C++:**

```
/*
* // This is the Master's API interface.
* // You should not implement it, or speculate about its implementation
* class Master {
* public:
* int guess(string word);
* };
*/
class Solution {
public:
void findSecretWord(vector<string>& words, Master& master) {

}
```

**Java:**

```
/**  
 * // This is the Master's API interface.  
 * // You should not implement it, or speculate about its implementation  
 * interface Master {  
 *     public int guess(String word) {}  
 * }  
 */  
class Solution {  
    public void findSecretWord(String[] words, Master master) {  
  
    }  
}
```

**Python3:**

```
# """  
# This is Master's API interface.  
# You should not implement it, or speculate about its implementation  
# """  
# class Master:  
#     def guess(self, word: str) -> int:  
  
class Solution:  
    def findSecretWord(self, words: List[str], master: 'Master') -> None:
```