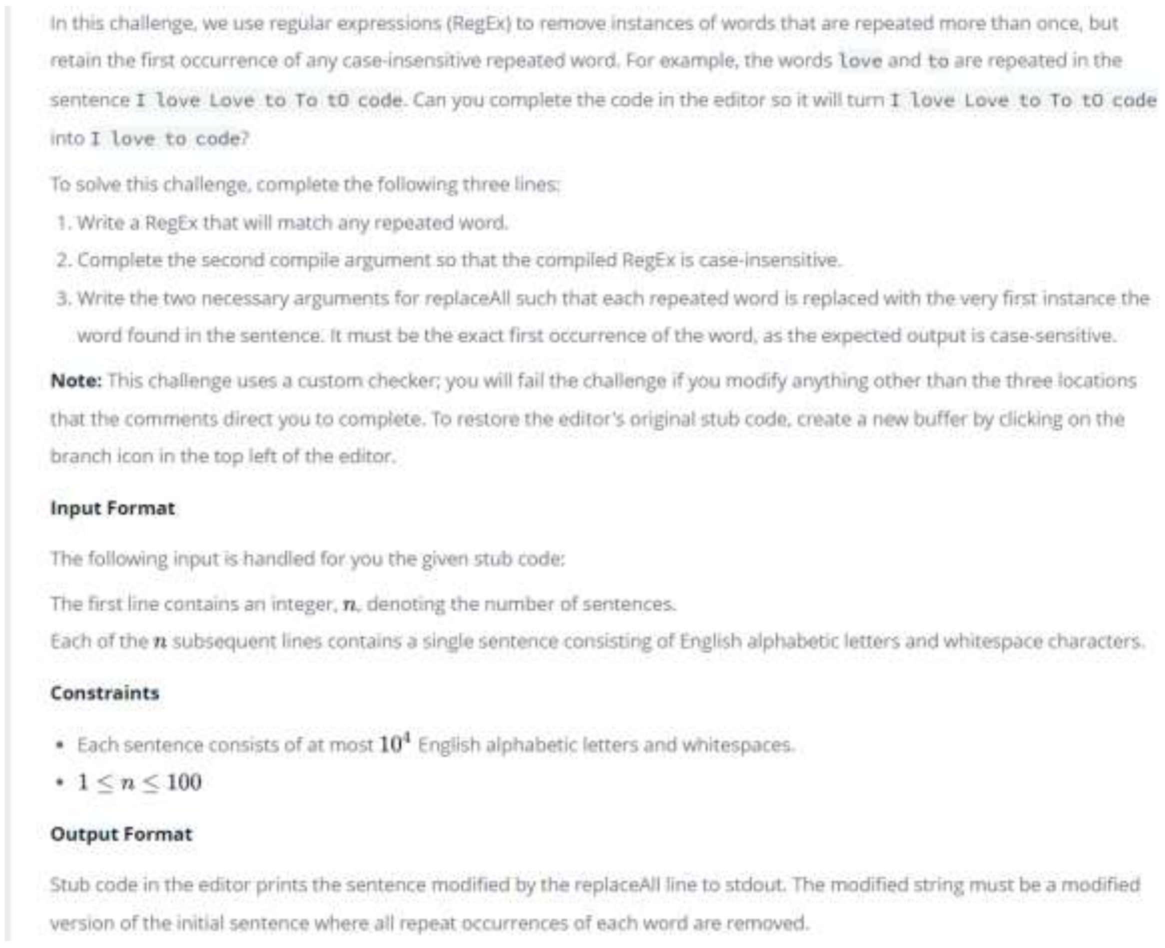
JAVA REGEX

Problem Statement



In this challenge, we use regular expressions (RegEx) to remove instances of words that are repeated more than once, but retain the first occurrence of any case-insensitive repeated word. For example, the words `love` and `to` are repeated in the sentence `I love Love to To tO code`. Can you complete the code in the editor so it will turn `I love Love to To tO code` into `I love to code`?

To solve this challenge, complete the following three lines:

1. Write a RegEx that will match any repeated word.
2. Complete the second compile argument so that the compiled RegEx is case-insensitive.
3. Write the two necessary arguments for replaceAll such that each repeated word is replaced with the very first instance the word found in the sentence. It must be the exact first occurrence of the word, as the expected output is case-sensitive.

Note: This challenge uses a custom checker; you will fail the challenge if you modify anything other than the three locations that the comments direct you to complete. To restore the editor's original stub code, create a new buffer by clicking on the branch icon in the top left of the editor.

Input Format

The following input is handled for you the given stub code:

The first line contains an integer, n, denoting the number of sentences.
Each of the n subsequent lines contains a single sentence consisting of English alphabetic letters and whitespace characters.

Constraints

- Each sentence consists of at most 10^4 English alphabetic letters and whitespaces.
- $1 \leq n \leq 100$

Output Format

Stub code in the editor prints the sentence modified by the replaceAll line to stdout. The modified string must be a modified version of the initial sentence where all repeat occurrences of each word are removed.

Code

```
import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;


public class DuplicateWords {
  public static void main(String[] args) {


        String regex = "\\b(\\w+)(?:\\W+\\1\\b)+";

        Pattern p = Pattern.compile(regex, Pattern.CASE_INSENSITIVE);


        Scanner in = new Scanner(System.in);
        int numSentences = Integer.parseInt(in.nextLine());

        while (numSentences-- > 0) {
            String input = in.nextLine();

            Matcher m = p.matcher(input);

            // Check for subsequences of input that match the compiled pattern
            while (m.find()) {
                input = input.replaceAll(m.group(), m.group(1));
            }
                        System.out.println(input);
        }

        in.close();
    }
}
```