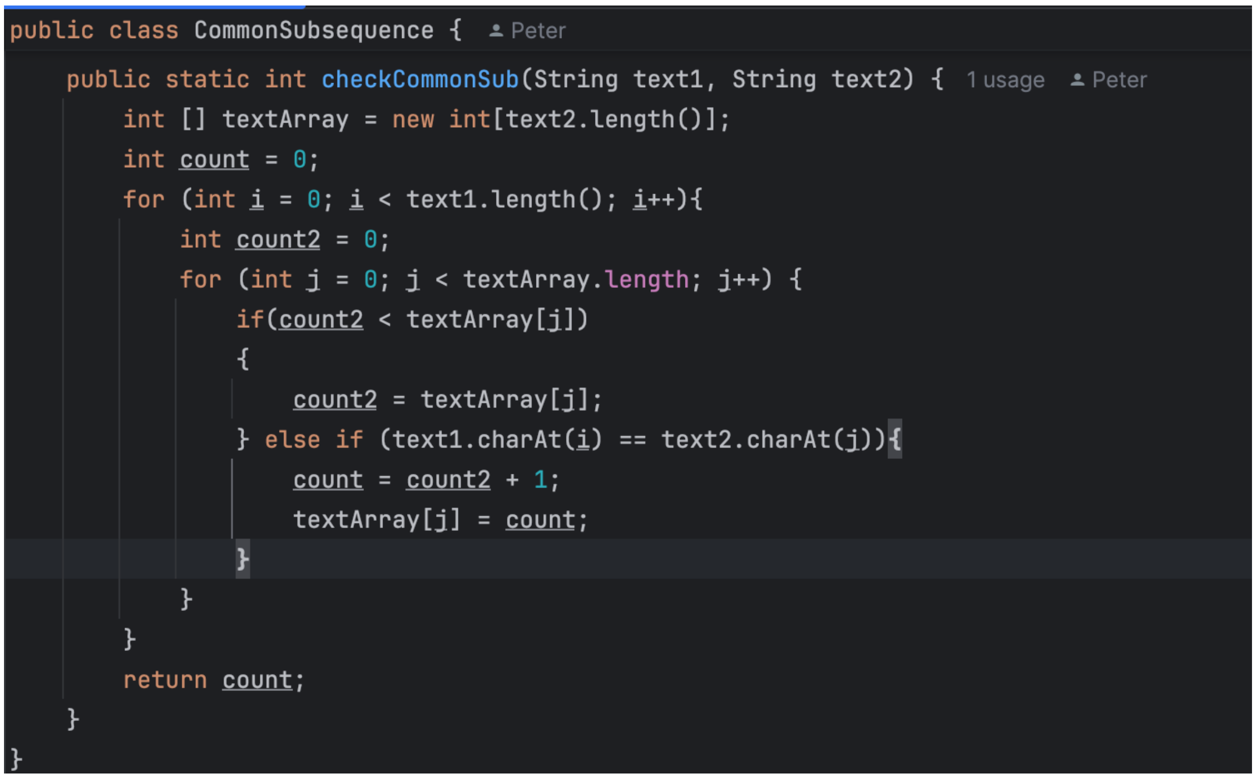
Time Complexity for all problems

#1.



```java
public class CommonSubsequence {  Peter
    public static int checkCommonSub(String text1, String text2) {  1 usage  Peter
        int [] textArray = new int[text2.length()];
        int count = 0;
        for (int i = 0; i < text1.length(); i++){
            int count2 = 0;
            for (int j = 0; j < textArray.length; j++) {
                if(count2 < textArray[j])
                {
                    count2 = textArray[j];
                } else if (text1.charAt(i) == text2.charAt(j)){
                    count = count2 + 1;
                    textArray[j] = count;
                }
            }
        }
        return count;
    }
}
```

When we take a look at this, this is indicating a nested loop, we can see that creating textArray and count is O(1) and return count is also O(1). When we look inside the inner loop, The inner loop iterates over each character in text2 (because the length of textArray is declared to be the length of text2 String). This loop runs text2.length() times, so the time complexity is O(n), where m is the length of text2. The outter loop iterating over each character in text1, so the time complexity is O(m). The reason it is n and m is because the text1 and text2 is independent and could have different length.

Big O is the worse case which is O(m x n). (upper bound)

In the best case, the loops still need to check each character, so the best-case time complexity is also Ω(n x m).

#2

Since the first code has a exact theta O(n x m), we can say that the second problem would have the same time complexity. Because creating a new array to store the letter are O(1)