Status	Finished
Started	Wednesday, 20 November 2024, 7:41 PM
Completed	Wednesday, 20 November 2024, 8:13 PM
Duration	32 mins 12 secs

Question **1**

Correct

Marked out of 5.00

Write a function that takes an input String (sentence) and generates a new String (modified sentence) by reversing the words in the original String, maintaining the words position.

In addition, the function should be able to control the reversing of the case (upper or lowercase) based on a case_option parameter, as follows:

If case_option = 0, normal reversal of words i.e., if the original sentence is "Wipro TechNologies BangaLore", the new reversed sentence should be "orpiW seigoloNhceT eroLagnaB".

If case_option = 1, reversal of words with retaining position's case i.e., if the original sentence is "Wipro TechNologies BangaLore", the new reversed sentence should be "Orpiw SeigOlonhcet ErolaGnab".

Note that positions 1, 7, 11, 20 and 25 in the original string are uppercase W, T, N, B and L.

Similarly, positions 1, 7, 11, 20 and 25 in the new string are uppercase O, S, O, E and G.

NOTE:

1. Only space character should be treated as the word separator i.e., "Hello World" should be treated as two separate words, "Hello" and "World". However, "Hello,World", "Hello;World", "Hello-World" or "Hello/World" should be considered as a single word.

2. Non-alphabetic characters in the String should not be subjected to case changes. For example, if case option = 1 and the original sentence is "Wipro TechNologies, Bangalore" the new reversed sentence should be "Orpiw ,seiGolonhceT Erolagnab". Note that comma has been treated as part of the word "Technologies," and when comma had to take the position of uppercase T it remained as a comma and uppercase T took the position of comma. However, the words "Wipro and Bangalore" have changed to "Orpiw" and "Erolagnab".

3. Kindly ensure that no extra (additional) space characters are embedded within the resultant reversed String.

Examples:

S. No.	input1	input2	output
1	Wipro Technologies Bangalore	0	orpiW seigolonhceT erolagnaB
2	Wipro Technologies, Bangalore	0	orpiW ,seigolonhceT erolagnaB
3	Wipro Technologies Bangalore	1	Orpiw Seigolonhcet Erolagnab
4	Wipro Technologies, Bangalore	1	Orpiw ,seigolonhceT Erolagnab

For example:

Input	Result
`Wipro Technologies Bangalore` `0`	`orpiW seigolonhceT erolagnaB`
`Wipro Technologies, Bangalore` `0`	`orpiW ,seigolonhceT erolagnaB`
`Wipro Technologies Bangalore` `1`	`Orpiw Seigolonhcet Erolagnab`
`Wipro Technologies, Bangalore` `1`	`Orpiw ,seigolonhceT Erolagnab`

Answer: (penalty regime: 0 %)

```
import java.util.Scanner;

public class WordReverser {
    public static String reverseWordsWithCase(String sentence, int caseOption) {
        // Split the sentence into words based on spaces
        String[] words = sentence.split(" ");

        // StringBuilder to store the result
        StringBuilder result = new StringBuilder();

        // Process each word
        for (String word : words) {
            // Reverse the word
            String reversedWord = new StringBuilder(word).reverse().toString();

            if (caseOption == 0) {
                // If caseOption is 0, no case conversion, just reverse the word
```

```
                result.append(reversedWord).append(" ");
            } else if (caseOption == 1) {
                // If caseOption is 1, adjust the case while maintaining original letter positions
                result.append(applyCaseConversion(reversedWord, word)).append(" ");
            }
        }

        // Remove the trailing space and return the result
        return result.toString().trim();
    }

    private static String applyCaseConversion(String reversedWord, String originalWord) {
        // StringBuilder to store the adjusted word
        StringBuilder adjustedWord = new StringBuilder();

        // Iterate over each character in the reversed word
        for (int i = 0; i < reversedWord.length(); i++) {
            char reversedChar = reversedWord.charAt(i);
            char originalChar = originalWord.charAt(i);

            if (Character.isLowerCase(originalChar)) {
                // If the original character was lowercase, the reversed character should be lowercase
                adjustedWord.append(Character.toLowerCase(reversedChar));
            } else if (Character.isUpperCase(originalChar)) {
                // If the original character was uppercase, the reversed character should be uppercase
                adjustedWord.append(Character.toUpperCase(reversedChar));
            } else {
                // Non-alphabetic characters remain unchanged
                adjustedWord.append(reversedChar);
            }
        }

        return adjustedWord.toString();
    }

```

	Input	Expected	Got	
✓	Wipro Technologies Bangalore 0	orpiW seigolonhceT erolagnaB	orpiW seigolonhceT erolagnaB	✓
✓	Wipro Technologies, Bangalore 0	orpiW ,seigolonhceT erolagnaB	orpiW ,seigolonhceT erolagnaB	✓
✓	Wipro Technologies Bangalore 1	Orpiw Seigolonhcet Erolagnab	Orpiw Seigolonhcet Erolagnab	✓
✓	Wipro Technologies, Bangalore 1	Orpiw ,seigolonhceT Erolagnab	Orpiw ,seigolonhceT Erolagnab	✓

Passed all tests! ✓

Question **2**

Correct

Marked out of 5.00

You are provided with a string which has a sequence of 1's and 0's.

This sequence is the encoded version of a English word. You are supposed write a program to decode the provided string and find the original word.

Each alphabet is represented by a sequence of 0s.

This is as mentioned below:

Z : 0

Y : 00

X : 000

W : 0000

V : 00000

U : 000000

T : 0000000

and so on upto A having 26 0's (00000000000000000000000000).

The sequence of 0's in the encoded form are separated by a single 1 which helps to distinguish between 2 letters.

Example 1:

input1: 010010001

The decoded string (original word) will be: ZYX

Example 2:

input1: 00001000000000000000001000000000001000000000100000000000001

The decoded string (original word) will be: WIPRO

Note: The decoded string must always be in UPPER case.

For example:

Input	Result
010010001	ZYX
00001000000000000000001000000000001000000000100000000000001	WIPRO

Answer: (penalty regime: 0 %)

```
import java.util.Scanner;
public class DecodeString {
    public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
String encodedString = scanner.nextLine();
StringBuilder decodedString = new StringBuilder();
int count = 0;
for (int i = 0; i < encodedString.length(); i++) {
    if (encodedString.charAt(i) == '0') { count++;
} else {
    char decodedChar = (char) ('Z' - count + 1);
    decodedString.append(decodedChar);
    count = 0;
}
}
System.out.println(decodedString.toString()); }
}
```

	Input	Expected	Got	
✓	010010001	ZYX	ZYX	✓
✓	00001000000000000000001000000000001000000000100000000000001	WIPRO	WIPRO	✓

Passed all tests! ✓

Question **3**

Correct

Marked out of 5.00

Given two char arrays input1[] and input2[] containing only lower case alphabets, extracts the alphabets which are present in both arrays (common alphabets).

Get the ASCII values of all the extracted alphabets.

Calculate sum of those ASCII values. Lets call it sum1 and calculate single digit sum of sum1, i.e., keep adding the digits of sum1 until you arrive at a single digit.

Return that single digit as output.

Note:

1. Array size ranges from 1 to 10.
2. All the array elements are lower case alphabets.
3. Atleast one common alphabet will be found in the arrays.

Example 1:

input1: {'a', 'b', 'c'}

input2: {'b', 'c'}

output: 8

Explanation:

'b' and 'c' are present in both the arrays.

ASCII value of 'b' is 98 and 'c' is 99.

98 + 99 = 197

1 + 9 + 7 = 17

1 + 7 = 8

For example:

Input	Result
a b c b c	8

Answer: (penalty regime: 0 %)

```
import java.util.HashSet; import java.util.Set; public class CommonAlphabetSum {
public static int singleDigitSum(int num) { int sum = 0;
while (num > 0) {
sum += num % 10;
num /= 10;
}
if (sum > 9) { return singleDigitSum(sum); }
return sum;
}
public static int calculateCommonAlphabetSum(char[] input1, char[] input2) { Set<Character> set1 = new HashSet<>()
}
int sum = 0; for (char c : input2) {
if (set1.contains(c)) { sum += c;
}
}
return singleDigitSum(sum);
}
public static void main(String[] args) { char[] input1 = {'a', 'b', 'c'}; char[] input2 = {'b', 'c', 'd'};
int result = calculateCommonAlphabetSum(input1, input2); System.out.println(result); }
}
```

	Input	Expected	Got	
✓	a b c b c	8	8	✓

Passed all tests! ✓

◄ Lab-12-MCQ

Jump to...

Identify possible words ►