


```

22         char reversedChar = reversedWord.charAt(i);
23
24         // If the character is alphabetic, apply case transformation
25         if (Character.isAlphabetic(originalChar)) {
26             if (Character.isUpperCase(originalChar)) {
27                 caseReversedWord.append(Character.toUpperCase(reversedChar));
28             } else {
29                 caseReversedWord.append(Character.toLowerCase(reversedChar));
30             }
31         } else {
32             // If it's a non-alphabetic character, keep it as is
33             caseReversedWord.append(reversedChar);
34         }
35     }
36
37     result.append(caseReversedWord).append(" ");
38 } else {
39     // Normal case, just append the reversed word
40     result.append(reversedWord).append(" ");
41 }
42 }
43
44 // Remove the extra space at the end
45 return result.toString().trim();
46 }
47
48 public static void main(String[] args) {
49     // Create a scanner to read input from user
50     Scanner sc = new Scanner(System.in);
51
52

```

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

Passed all tests!

Question **2**

Correct

Marked out of 5.00

🚩 Flag question

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 (000000000000000000000000000000).

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: 0000100000000000000000001000000000001000000000100000000000001

The decoded string (original word) will be: WIPRO

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

For example:

Input	Result
010010001	ZYX
0000100000000000000000001000000000001000000000100000000000001	WIPRO

Answer: (penalty regime: 0 %)

```

1 import java.util.Scanner;
2
3 public class DecodeString {
4
5     // Function to decode the string
6     public static String decodeString(String input) {
7         // Split the input string by '1', as '1' is the delimiter
8         String[] parts = input.split("1");
9         StringBuilder result = new StringBuilder();
10
11         // Iterate over each part
12         for (String part : parts) {
13             // Skip empty parts (which could appear due to multiple consecutive '1's)
14             if (!part.isEmpty()) {
15                 int length = part.length();
16
17                 // Calculate the letter corresponding to the length of 0's
18                 // 'Z' corresponds to 1 zero, 'Y' to 2 zeros, ..., 'A' corresponds to 26 zeros
19                 char letter = (char) ('Z' - (length - 1));
20                 result.append(letter);
21             }
22         }
23
24         return result.toString();
25     }
26
27     public static void main(String[] args) {
28         // Create scanner object to read input
29         Scanner sc = new Scanner(System.in);
30
31
32         String input = sc.nextLine();
33
34         // Decode the string and output the result
35         String decodedString = decodeString(input);
36         System.out.println(decodedString);
37
38         sc.close();
39     }
40 }

```

[illegible]

Passed all tests!

Question 3

Correct

Marked out of 5.00

Flag question

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	8
b c	

Answer: (penalty regime: 0 %)

```
1 public class CommonCharsAsciiSum{
2     public static int calculatesingledigitsum(char[] input1,char[] input2){
3         StringBuilder commonChars=new StringBuilder();
4         for(char c1:input1){
5             for(char c2:input2){
6                 if(c1==c2){
7                     commonChars.append(c1);
8                     break;
9                 }
10            }
11        }
12        int asciisum=0;
13        for(int i=0;i<commonChars.length();i++){
14            asciisum+=commonChars.charAt(i);
15        }
16        while(asciisum>=10){
17            int tempsum=0;
18            while(asciisum>0){
19                tempsum+=asciisum%10;
20                asciisum=asciisum/10;
21            }
22            asciisum=tempsum;
23        }
24        return asciisum;
25    }
26    public static void main(String[] args){
27        char[] input1={'a','b','c'};
28        char[] input2={'b','c'};
29        System.out.println(calculatesingledigitsum(input1,input2));
30    }
31 }
```

	Input	Expected	Got	
	a b c	8	8	
	b c			

Passed all tests!

[Finish review](#)

[◀ Lab-12-MCQ](#)

Jump to...

[Identify possible words ▶](#)