Dashboard / My courses / CS23333-OOPUJ-2023 / Lab-12-Introduction to I/O, I/O Operations, Object Serialization / Lab-12-Logic Building

Status	Finished
Started	Sunday, 17 November 2024, 1:00 AM
Completed	Sunday, 17 November 2024, 1:15 AM
Duration	15 mins 21 secs

Question **1**

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: 00001000000000000000000010000000000010000000001000000000000l

The decoded string (original word) will be: WIPRO

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

For example:

Input	Result
`010010001`	`ZYX`
`00001000000000000000000100000000000100000000010000000000001`	`WIPRO`

Answer: (penalty regime: 0 %)

```java
import java.util.Scanner;

public class DecodeString {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Read the encoded string input
        String encodedString = scanner.nextLine();

        // Split the encoded string into parts using '1' as a delimiter
        String[] parts = encodedString.split("1");

        StringBuilder decodedWord = new StringBuilder();

        // Decode each part
        for (String part : parts) {
            if (!part.isEmpty()) { // Ignore empty parts due to leading/trailing or consecutive 1s
                int zeroCount = part.length();
                char letter = (char) ('Z' - (zeroCount - 1)); // Calculate the character
                decodedWord.append(letter); // Append the letter to the result
            }
        }

        // Print the decoded word
        System.out.println(decodedWord.toString());

        scanner.close();
    }
}

```

	Input	Expected	Got	
✓	010010001	ZYX	ZYX	✓
✓	00001000000000000000000100000000000100000000010000000000001	WIPRO	WIPRO	✓

Passed all tests! ✓

Question **2**

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 %)

```java
import java.util.Scanner;

public class ReverseWordsWithCaseOption {
    public static String reverseWords(String sentence, int caseOption) {
        String[] words = sentence.split(" "); // Split words by spaces
        StringBuilder result = new StringBuilder();

        for (String word : words) {
            StringBuilder reversedWord = new StringBuilder();

            // Reverse the word
            for (int i = word.length() - 1; i >= 0; i--) {
                reversedWord.append(word.charAt(i));
            }

            // Apply case adjustment for caseOption == 1
            if (caseOption == 1) {
                for (int i = 0; i < reversedWord.length(); i++) {
                    char originalChar = word.charAt(i);
                    char reversedChar = reversedWord.charAt(i);

                    if (Character.isUpperCase(originalChar)) {
                        reversedWord.setCharAt(i, Character.toUpperCase(reversedChar));
                    } else if (Character.isLowerCase(originalChar)) {
                        reversedWord.setCharAt(i, Character.toLowerCase(reversedChar));
                    }
                }
            }

            // Append the processed word to the result
            result.append(reversedWord).append(" ");
        }

        // Trim any trailing space and return
        return result.toString().trim();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Input the sentence

        String sentence = sc.nextLine();

        // Input the case option

        int caseOption = sc.nextInt();

        // Get the result and print
        String result = reverseWords(sentence, caseOption);
        System.out.println(result);

```

	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 **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 %)

```java
import java.util.HashSet;
import java.util.Scanner;

public class CommonAlphabetAsciiSum {

    // Function to calculate the single-digit sum
    public static int calculateSingleDigitSum(int sum) {
        while (sum > 9) {
            int digitSum = 0;
            while (sum > 0) {
                digitSum += sum % 10;
                sum /= 10;
            }
            sum = digitSum;
        }
        return sum;
    }

    // Main function to calculate the result
    public static int commonAlphabetAsciiSum(char[] input1, char[] input2) {
        HashSet<Character> set1 = new HashSet<>();
        HashSet<Character> commonSet = new HashSet<>();

        // Add elements of input1 to set1
        for (char ch : input1) {
            set1.add(ch);
        }

        // Find common elements between input1 and input2
        for (char ch : input2) {
            if (set1.contains(ch)) {
                commonSet.add(ch);
            }
        }

        // Calculate the ASCII sum of common elements
        int sum = 0;
        for (char ch : commonSet) {
            sum += (int) ch;
        }

        // Return the single-digit sum
        return calculateSingleDigitSum(sum);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Input for array 1

        String[] input1Str = sc.nextLine().split(" ");
        char[] input1 = new char[input1Str.length];
```

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

Passed all tests! ✓

◄ Lab-12-MCQ

Jump to...

Identify possible words ►