WEEK-12

ROLL NO:230701233

1. 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.

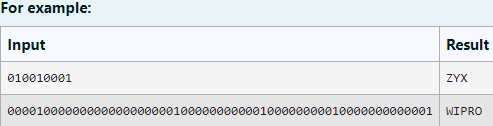
Example 1:

input1: 010010001

The decoded string (original word) will be: ZYX Example 2:

input1: 00001000000000000000000100000000000100000000010000000000001

The decoded string (original word) will be: WIPRO



CODE:

import java.util.Scanner;

public class DecodeBinaryString {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); String input = scanner.nextLine();

String decodedString = decodeString(input); System.out.println( decodedString);

scanner.close();

}

private static String decodeString(String input) { String[] zeroSequences = input.split("1");

StringBuilder decodedWord = new StringBuilder(); for (String zeroSequence : zeroSequences) {

int length = zeroSequence.length(); if (length > 0) {

char letter = (char) ('Z' - (length - 1)); decodedWord.append(letter);

}

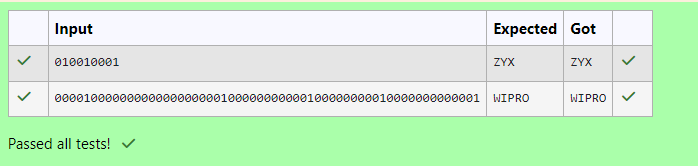
}

return decodedWord.toString();

}

}

OUTPUT:

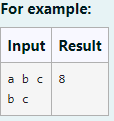


1. Given two char arrays input1[] and input2[] containing only lower case alphabets, extracts the alphabets which are present in both arrays (common alphabets).

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.



Code:

import java.util.HashSet; import java.util.Set;

public class CommonAlphabetsSum {

public static int getSingleDigitSum(int sum) { while (sum >= 10) {

sum = sumOfDigits(sum);

}

return sum;

}

private static int sumOfDigits(int num) { int sum = 0;

while (num > 0) {

sum += num % 10; num /= 10;

}

return sum;

}

public static int commonAlphabetsSingleDigitSum(char[] input1, char[] input2) { Set<Character> set1 = new HashSet<>();

Set<Character> set2 = new HashSet<>(); for (char c : input1) {

set1.add(c);

}

for (char c : input2) { set2.add(c);

}

set1.retainAll(set2); int asciiSum = 0;

for (char c : set1) { asciiSum += (int) c;

}

return getSingleDigitSum(asciiSum);

}

public static void main(String[] args) { char[] input1 = {'a', 'b', 'c'};

char[] input2 = {'b', 'c'};

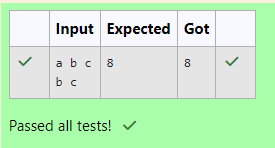
int result = commonAlphabetsSingleDigitSum(input1, input2);

System.out.println(result);

}

}

Output:



1. 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:

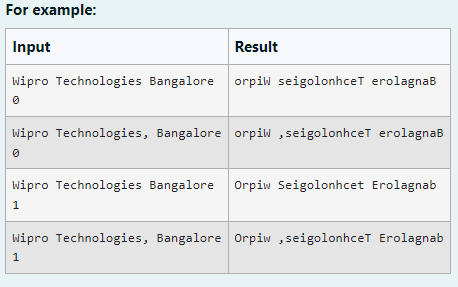
If case\_option = 0, normal reversal of words i.e., if the original sentence is “Wipro

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.



CODE:

import java.util.Scanner;

public class SentenceReverser {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); String sentence = scanner.nextLine();

int caseOption = scanner.nextInt();

String result = reverseWords(sentence, caseOption); System.out.println( result);

scanner.close();

}

/\*\*

* This function takes a sentence and reverses each word as per the case\_option.

\*

* @param sentence The original sentence to process.
* @param caseOption The option to handle case sensitivity (0 for normal, 1 for case- sensitive).
* @return The modified sentence with reversed words.

\*/

public static String reverseWords(String sentence, int caseOption) { String[] words = sentence.split(" ");

StringBuilder result = new StringBuilder(); for (int w = 0; w < words.length; w++) {

String word = words[w];

StringBuilder reversedWord = new StringBuilder(word.length()); for (int i = 0; i < word.length(); i++) {

char originalChar = word.charAt(i);

char reversedChar = word.charAt(word.length() - 1 - i);

if (caseOption == 1 && Character.isLetter(originalChar)) { if (Character.isUpperCase(originalChar)) {

reversedChar = Character.toUpperCase(reversedChar);

} else {

reversedChar = Character.toLowerCase(reversedChar);

}

}

reversedWord.append(reversedChar);

}

if (w > 0) { result.append(" ");

}

result.append(reversedWord);

}

return result.toString();

}

}

OUTPUT:

