**WEEK-12**

**Question 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**

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

**For example:**

| **Input** | **Result** |
| --- | --- |
| **010010001** | **ZYX** |
| **00001000000000000000000100000000000100000000010000000000001** | **WIPRO** |

**Program:**

**import java.util.Scanner;**

**public class Decoder {**

**// Method to decode the sequence**

**public static String decode(String input) {**

**// Split the input by '1' to separate the sequences of 0's**

**String[] sequences = input.split("1");**

**// StringBuilder to build the decoded word**

**StringBuilder decodedWord = new StringBuilder();**

**// Iterate over each sequence**

**for (String sequence : sequences) {**

**// If the sequence is not empty (it could be empty due to split)**

**if (!sequence.isEmpty()) {**

**int length = sequence.length();**

**// The letter corresponding to the sequence length**

**// 'Z' corresponds to length 1, 'Y' to length 2, ..., 'A' to length 26**

**char letter = (char) ('Z' - (length - 1));**

**// Append the letter to the decoded word**

**decodedWord.append(letter);**

**}**

**}**

**return decodedWord.toString();**

**}**

**public static void main(String[] args) {**

**// Example input 1**

**Scanner scanner = new Scanner(System.in);**

**String input1= scanner.nextLine();**

**System.out.println(decode(input1)); // Output: ZYX**

**}**

**}**

****

**Question 2**

**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:**

****

**Program:**

**import java.util.Scanner;**

**public class SentenceReverser {**

**public static String reverseWords(String sentence, int caseOption) {**

**// Split the sentence into words based on spaces**

**String[] words = sentence.split(" ");**

**StringBuilder result = new StringBuilder();**

**for (int i = 0; i < words.length; i++) {**

**String reversedWord = reverseWord(words[i], caseOption);**

**result.append(reversedWord);**

**// Add space after each word except the last one**

**if (i < words.length - 1) {**

**result.append(" ");**

**}**

**}**

**return result.toString();**

**}**

**private static String reverseWord(String word, int caseOption) {**

**StringBuilder reversed = new StringBuilder();**

**// Reverse the word**

**for (int i = word.length() - 1; i >= 0; i--) {**

**reversed.append(word.charAt(i));**

**}**

**// If caseOption is 1, adjust the case based on the original positions**

**if (caseOption == 1) {**

**char[] resultChars = reversed.toString().toCharArray();**

**for (int i = 0; i < word.length(); i++) {**

**if (Character.isLetter(word.charAt(i))) {**

**if (Character.isUpperCase(word.charAt(i))) {**

**resultChars[i] = Character.toUpperCase(resultChars[i]);**

**} else {**

**resultChars[i] = Character.toLowerCase(resultChars[i]);**

**}**

**}**

**}**

**return new String(resultChars);**

**}**

**// If caseOption is 0, return the reversed word as is**

**return reversed.toString();**

**}**

**public static void main(String[] args) {**

**Scanner scanner = new Scanner(System.in);**

**// Prompt user for sentence input**

**String sentence = scanner.nextLine();**

**// Prompt user for case option input**

**int caseOption = scanner.nextInt();**

**// Output the result**

**String result = reverseWords(sentence, caseOption);**

**System.out.println(result);**

**scanner.close();**

**}**

**}**

****

**Question 3**

**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** |

**Program:**

**import java.util.HashSet;**

**import java.util.Scanner;**

**public class CommonAlphabetSum {**

**public static int calculateSingleDigitSum(int sum) {**

**// Keep adding digits until the sum is a single digit**

**while (sum > 9) {**

**int temp = 0;**

**while (sum != 0) {**

**temp += sum % 10;**

**sum /= 10;**

**}**

**sum = temp;**

**}**

**return sum;**

**}**

**public static int findCommonAlphabetSum(char[] input1, char[] input2) {**

**// Convert the first input array to a set to get unique characters**

**HashSet<Character> set1 = new HashSet<>();**

**for (char c : input1) {**

**set1.add(c);**

**}**

**// Sum the ASCII values of characters present in both arrays**

**int sum = 0;**

**for (char c : input2) {**

**if (set1.contains(c)) {**

**sum += (int) c;**

**}**

**}**

**// Calculate the single-digit sum**

**return calculateSingleDigitSum(sum);**

**}**

**public static void main(String[] args) {**

**Scanner scanner = new Scanner(System.in);**

**// Input for the first array**

**String input1Str = scanner.nextLine();**

**char[] input1 = input1Str.replace(" ", "").toCharArray();**

**// Input for the second array**

**String input2Str = scanner.nextLine();**

**char[] input2 = input2Str.replace(" ", "").toCharArray();**

**// Calculate and print the result**

**int result = findCommonAlphabetSum(input1, input2);**

**System.out.println(result);**

**scanner.close();**

**}**

**}**

****