Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2016

Etnetera Brevity Challenge

`brevity.c`, `brevity.cpp`, `brevity.c11`, `Brevity.java`, `brevity.py`

Your task is to implement a decoder of the famous Morse alphabet. As most of you know, the Morse code represents characters as variable-length sequences of short and long signals ("beeps"), often written as dots and dashes. For those who do not remember their scouting years, the following table shows the Morse code sequences for all letters:

A	`.-`	E	`.`	I	`..`	M	`--`	Q	`--.-`	U	`..-`	Y	`-.--`
B	`-...`	F	`..-.`	J	`.---`	N	`-.`	R	`.-.`	V	`...-`	Z	`--..`
C	`-.-.`	G	`--.`	K	`-.-`	O	`---`	S	`...`	W	`.--`		
D	`-..`	H	`....`	L	`.-..`	P	`.--.`	T	`-`	X	`-..-`		

If more letters are to be transferred, they are separated by a short pause, typically written as a slash. A space between words is represented by an even longer pause, written as two slashes.

Input Specification

The input contains several test cases. Each test case is specified on one line with at most 1000 characters, which describes a valid Morse code transmission. Specifically:

- The line consists only of dashes ("`-`"), dots ("`.`"), and slashes ("`/`").
- There is at least one character.
- The first and last characters will never be a slash.
- There will never be more than two slashes together.
- Each non-empty sequence between two slashes contains a valid Morse code of one letter.

Output Specification

For each test case, print one line containing the decoded message in uppercase letters.

Sample Input

```
.-/-.-./--
-.-./-/..-//---/.--././-.
./-/-././-/./.-./.-//-.../.-././.../-/../-/-.--//-.-./..../.-/.-../.-.././-./--./.
```

Output for Sample Input

```
ACM
CTU OPEN
ETNETERA BREVITY CHALLENGE
```

Etnetera Brevity Challenge — Rules

To make this practice even more interesting, we have prepared a special challenge for you. Sort-of "a contest inside the Contest". The goal of the *Etnetera Brevity Challenge* is to write as short code as possible. On the other side of this paper, you will find the statement of a problem. The shortest correct solution of the problem wins our challenge — and will be awarded!

Your solution must meet the following requirements to be accepted to the Challenge:

- It must correctly solve the problem, i.e., be judged as "correct" by the contest system. (It is possible to submit a correct solution repeatedly, the best attempt counts.)
- It must be submitted before the end of the Practice Session.
- It must be in C, C++, or Java. Sorry, Python solutions are not allowed this time.

To evaluate the length of solutions, the following rules will be applied:

- The length is measured as the number of individual *characters* of your source code.
- Any syntactically-irrelevant whitespace will not be counted. Which means, you do not have to compress spaces and newlines: if some whitespace may be deleted and the program still works, it will be deleted.
- We will *not* count the following lines and also two closing parentheses ("}").

```
#include <*>
int main() {
    char input[1000]; while (scanf("%s", input)>0) {
        printf("%s\n", demorse(input));
    return 0;

using namespace std;
    string input; while (cin >> input) {
        cout << demorse(input) << endl;

import java.*;
public class Brevity {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        Brevity inst = new Brevity();
        String line; while ((line = br.readLine()) != null)
            System.out.println(inst.demorse(line));
```

As the evaluation is automatic, it is necessary to include the above lines exactly as given (ignoring any whitespace) and whole lines only. The purpose is to avoid optimizing input processing and concentrate on the implementation of the `demorse` method/function only.

The goal of this Challenge is to write the shortest possible implementation, not to find leaks in these rules. If you try to abuse any rule, some penalty may be awarded, at the dicretion of the organizers.

In the case of a tie, the organizers are solely responsible for specifying additional criteria to determine the winner.