

[Practice](#)[Compete](#)[Jobs](#)[Rank](#)[Leaderboard](#)

Buritomath ▾

[Dashboard](#) > [Functional Programming](#) > [Recursion](#) > Prefix CompressionBadge Progress [\(Details\)](#)

Points: 143.00 Rank: 4990

Prefix Compression



by abhiranjan

Problem

Submissions

Leaderboard

Discussions

Editorial

You are in charge of data transfer between two Data Centers. Each set of data is represented by a pair of strings. Over a period of time you have observed a trend: most of the times both strings share some prefix. You want to utilize this observation to design a data compression algorithm which will be used to reduce amount of data to be transferred.

You are given two strings, x and y , representing the data, you need to find the longest common prefix (p) of the two strings. Then you will send substring p , x' and y' , where x' and y' are the substring left after stripping p from them.

For example, if $x = "abcdefpr"$ and $y = "abcpqr"$, then $p = "abc"$, $x' = "defpr"$, $y' = "pqr"$.

Input Format

The first line contains a single string denoting x .

The second line contains a single string denoting y .

Constraints

- x and y will contain only lowercase Latin characters ('a'-'z').
- $1 \leq \text{length}(x), \text{length}(y) \leq 10^5$

Output Format

In first line, print the length of substring p , followed by prefix p . In second line, print the length of substring x' , followed by substring x' . Similarly in third line, print the length of substring y' , followed by substring y' .

Sample Input 0

```
abcdefpr
abcpqr
```

Sample Output 0

```
3 abc
5 defpr
3 pqr
```

Sample Input 1

```
kitkat
kit
```

Sample Output 1

```
3 kit
3 kat
0
```

Sample Input 2

```
puppy
puppy
```

Sample Output 2

```
5 puppy
0
0
```

Explanation

Sample Case 0:

Already explained above in the problem statement.

Sample Case 1:

$p = \text{"kit"}$, which is also y . So x' will be "kat" and y' will be an empty string.

Sample Case 2:

Because both strings are the same, the prefix will cover both the strings. Thus, x' and y' will be empty strings.

[f](#) [t](#) [in](#)

Submissions: 1733

Max Score: 10


Difficulty: Easy

Rate This Challenge:

☆☆☆☆☆

[More](#)

If you cannot find your favorite language here, please check out the [algorithm domain](#). We support over 40 popular languages.

Current Buffer (saved locally, editable)  

Racket



```
1 #lang racket
2 ; Enter your code here. Read input from STDIN. Print output to STDOUT
3
4 (define x (read-line))
5 (define y (read-line))
6
7 (define (getprefix-iter x y string)
8   (cond ((or (eq? x '()) (eq? y '()) (not (eq? (car x) (car y)))) (reverse string))
9         (else (getprefix-iter (cdr x) (cdr y) (cons (car x) string)))))
10
11 (define common-prefix (list->string (getprefix-iter (string->list x) (string->list y) '())))
12 (define common-prefix-length (string-length common-prefix))
13
14 (displayln (string-append (number->string common-prefix-length) " " common-prefix " "))
15
16 (define substrx (substring x common-prefix-length))
17 (define substry (substring y common-prefix-length))
18
19 (displayln (string-append (number->string (- (string-length x) common-prefix-length)) " " substrx))
20 (displayln (string-append (number->string (- (string-length y) common-prefix-length)) " " substry))
```

Line: 1 Col: 1

 [Upload Code as File](#) ☐ [Test against custom input](#)

[Run Code](#)

[Submit Code](#)

Copyright © 2017 HackerRank. All Rights Reserved

Join us on IRC at [#hackerrank](#) on freenode for hugs or bugs.

[Contest Calendar](#) | [Interview Prep](#) | [Blog](#) | [Scoring](#) | [Environment](#) | [FAQ](#) | [About Us](#) | [Support](#) | [Careers](#) | [Terms Of Service](#) | [Privacy Policy](#) | [Request a Feature](#)