A New Order

Nicholas Newton does not using the names of students to do arrangements, since this can force students with names that start with z or a to always be at the ends, which means they have less people to talk with, while students with names closer to the middle are always half way to either end, never having the ability to leave early. Some times student can be arranged from a to z; other times students can be arranged from z to a. Nicholas wants to end this sort of discrimination. To give students a break from the potential monotony Nicholas will rearrange an alphabet and sort the students using the new ordering scheme.

Problem

Nicholas will give a new ordering of the alphabet from first to last, and an unordered list of names. Print the names of the students after ordering them using the given alphabetical order.

Input Specification

The input will begin with 26 space separated, lower case, pairwise distinct, Latin characters. The letters will represent the desired alphabetical ordering of the list of names from first to last. The next line will contain a single number, **n** ($n < 100,000$), representing the number of names. The following n lines will each contain a single name as a lowercase string of at most 100 lowercase letters with no whitespace.

Output Specification

The output will consist of a **n** lines, each containing a string, representing the modified order of the students based on the given arrangements of letters.

Sample Input	Sample Output

`b j k v q z c d e t u h w x n o p f g l y m r s i a`
`5`
`jonah`
`peter`
`frank`
`john`
`adam`
	`john`
	`jonah`
	`peter`
	`frank`
	`adam`

`u l g q k z o d n v p y h a r x i c e f w m j b t s`
`9`
`arthur`
`selina`
`slade`
`bart`
`bruce`
`wally`
`diana`
`barry`
`billy`
	`diana`
	`arthur`
	`wally`
	`barry`
	`bart`
	`bruce`
	`billy`
	`slade`
	`selina`

`a b c d e f g h i j k l m n o p q r s t u v w x y z`
`1`
`jess`
	`jess`

Explanation

In the *first case* the letter j comes before a, f, or p. Thus John and Jonah must be first. Since they both begin with “jo” we need to look at the third letter of each string and use that to sort these two. The third letter is h versus n. The h occurs before the n in the given array, so John will be first. We can notice that a is the last letter of the array, so Adam must be last as he is the only a name. The letter p is just before the letter f in our array, so peter must come before frank. With this described ordering of names the only possible correct ordering is the one given.

John, Jonah, Peter, Frank, Adam

In the *second case* case the letter ‘d’ comes before ‘a’, ‘w’, ‘b’, and ‘s’, so Diana is the first name.

Diana								

The next letter in our permuation is ‘a’, so Arthur is next.

Diana	Arthur							

The letter ‘w’ occurs before ‘b’ and ‘s’, so wally is third.

Diana	Arthur	Wally						

Since ‘s’ is the last letter of the given permutation of characters, Selina and Slade will be last.

Diana	Arthur						(Slade, Selina)	

Slade will come before Selina, since ‘l’ is the second letter of our permuation only preceded by ‘u’ (and not ‘e’).

Diana	Arthur	Wally					Slade	Selina

With Barry, Bart, Billy, and Bruce we need to look at the second letter (and for Barry and Bart the forth character).

Diana	Arthur	Wally	(Barry, Bart, Billy, Bruce)				Slade	Selina

The letter ‘a’ comes before ‘i’ and ‘r’, so Bart and Barry will be before Billy and Bruce.

Diana	Arthur	Wally	(Bart, Barry)		(Billy, Bruce)		Slade	Selina

Bart is after Barry because in our permutation ‘t’ is after ‘r’.

Diana	Arthur	Wally	Barry	Bart	(Billy Bruce)		Slade	Selina

For Billy and Bruce we choose Bruce first since r is 2 characters ahead of ‘i’.

Diana	Arthur	Wally	Barry	Bart	Bruce	Billy	Slade	Selina

Grading Details

Read/Write from/to standard input/output – 10 points

Good comments, whitespace, and variable names – 15 points

No extra input output (e.g. input prompts, “Please enter the number of words”) – 10 points

Write a custom comparison method that takes in two strings and evaluates the ordering – 10 points

Write a sorting method to arrange the values – 5 points

Your program will be tested on 10 test cases – 5 points each

No points will be awarded to programs that do not compile using gcc -std=gnu11 (gnu “eleven”).

Sometimes a requested technique will be given, and solutions without the requested technique will have their maximum points total reduced. For this problem you must write a sorting method. ***Without a sorting method your program will earn at most 50 points!***

Any case that causes your program to return a non-zero error return code will be treated as completely wrong. Additionally any case that takes longer than the maximum allowed time (the max of {5 times my solution, 5 seconds}) will also be treated as wrong. You will most likely need to write a sort that runs in O(n log (n)) comparisons to finish all the test cases within the allotted time.

No partial credit will be awarded for an incorrect case.