ROT 13.5

A simple method for encrypting data is that of ROT 13. The method takes each latin letter of plain text and moves it to the next letter 13 times in latin alphabet (wrapping around if necessary). For those of you who are unaware the latin alphabet is the following

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

This means the letter ‘a’ would become the letter ‘n’. The word “hello” would become “uryyb”. To recover the original text you only need to apply ROT 13 to the encrypted text. The word “uryyb” under ROT 13 becomes “hello”. Due to the simplicity of the encryption this method is no longer used in practice.

Problem

For the first homework assignment of the semester you are to implement a similar encryption scheme called ROT 13.5. In this encryption scheme whenever you encrypt a vowel (a, e, i, o, u) as plain text, the next encrypted letter moves to the next letter 14 times in the latin alphabet. In all other cases a plain text letter will be moved to the next letter 13 times. This means that the word “hello” becomes “urzyb” under rot 13.5.

Input Specification

Input will start with a single positive integer n (n < 10). The next n lines contains a single word consisting of lower case latin letters (no more than 100 characters in length). No extra whitespace will be present.

Output Specification

For each word given in input your program is to output a line containing ONLY the word encrypted as ROT 13.5 with no extra whitespace or characters in lowercase latin characters.

Sample Input	Sample Output
2 hello world	urzyb jbfyq
1 urzyb	hfmlo
3 uqyyb pf v	hello cs i

Grading Details

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

Good comments, whitespace, and variable names – 15 points

Read in all the input – 5 points

No extra input output (e.g. input prompts, “Please enter the number of words”) – 10 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 use `scanf(“%s”, ...)` *to read in the words.* ***Without this 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, 10 seconds}) will also be treated as wrong.

No partial credit will be awarded for an incorrect case.