Roman Palindromes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Given the string S, consisting of letters '`I`', '`V`', '`X`', '`L`', '`C`', '`D`' and '`M`'. Your task is to split the string to the **minimum** number of strings that are:

- correctly written Roman numerals;
- palindromes.

The following table from Wikipedia displays how the Roman Numerals are written:

Individual decimal places	Thousands	Hundreds	Tens	Units
1	M	C	X	I
2	MM	CC	XX	II
3	MMM	CCC	XXX	III
4		CD	XL	IV
5		D	L	V
6		DC	LX	VI
7		DCC	LXX	VII
8		DCCC	LXXX	VIII
9		CM	XC	IX

Note that:

- The numerals for 4, 9, 40, 90, 400 and 900 are written using "subtractive notation" where the first symbol is subtracted from the larger one (for example, for 40 ("`XL`") '`X`' (10) is subtracted from '`L`' (50)). These are **the only** subtractive forms in standard use.
- A number containing several decimal digits is built by appending the Roman numeral equivalent for each, from highest to lowest.
- Any missing place (represented by a zero in the place-value equivalent) is omitted.
- The largest number that can be represented in the Roman notation is 3,999 (MMMCMXCIX).

Input

The first line of the input consists of one integer n — length of the string ($1 \le n \le 100\,000$).

The second line contains the string of length n, consisting of letters '`I`', '`V`', '`X`', '`L`', '`C`', '`D`' and '`M`'.

Output

In first line of the output print one integer k — the minimum number of correct roman palindromes that can form the given string being concatenated. Then print k lines, each line containing one string — a correct Roman numeral that is a palindrome, such as the concatenation of all k lines in the order of output is equal to the given string.

If there is more than one solution, print any of them.

Example

standard input	standard output
5 MMXXI	3 MM XX I