Problem F. Format

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

In C/C++, format "`%[..]`" is used by `scanf` to read a string consisting of characters defined by this format string.

The left bracket is followed by a sequence of characters and a right bracket. The sequence f of characters between the brackets defines a set of accepted characters. Only characters with ASCII codes from 32 (space) to 126 ('`~`') inclusive are acceptable there. In this problem, the characters '`^`', '`]`' and '`-`' are used in format only as control characters.

If the first character of sequence f is not a circumflex ('`^`'), then the sequence f defines the set of available characters s, and the function reads all characters up to the first character that is not in the set s or the end of the input. If the first character of sequence f is '`^`', then it must be followed by a **non-empty** sequence of characters defining the set of forbidden characters s, and the function reads all characters up to the first character in the set s or the end of the input.

A group of characters with sequential ASCII codes may be abbreviated as an *interval*: first character in the group, '`-`' (minus sign, ASCII code 45), last character in the group. For example, interval "`B-Q`" defines a set of uppercase English letters from '`B`' to '`Q`' inclusive.

A string is called *alphanumeric* if it consists only of upper- and lowercase English letters, digits and spaces. Given an alphanumeric string t, build the "`%[..]`" format which:

1. Accepts string t: when this string is given as input, scanf reads it fully.
2. Accepts the least possible number of other alphanumeric strings of length $|t|$.
3. Has the minimum possible length (as a string) among all format strings that conform to all the points above.
4. Is lexicographically smallest among all format strings that conform to all the points above.

Input

The first line of input contains one non-empty string t consisting only of upper- and lowercase English letters, digits and spaces. It is guaranteed that this string is no longer than 100 000 characters, does not have leading or trailing spaces, and does not have two or more consecutive spaces.

Output

Print the answer as a format string "`%[..]`" (without quotes).

Examples

standard input	standard output
bc0123456789ABCDEFxxyyzz	%[!-Fbcx-z]
012345678 abcdefABCDEF	%[^9G-Zg-z]
01234567 abcdefABCDEF	%[-7:-F[-f]

Note

Note that the format string may contain non-alphanumeric characters. For example, in sample 1, '`!`' (character with ASCII code 33) is used instead of '`0`' because the number of alphanumeric characters between '`!`' and '`F`' and between '`0`' and '`F`' is the same, but format string with '`!`' is lexicographically smaller.

String a is lexicographically smaller than string b if b is not a prefix of a and one of two conditions hold:

1. a is a prefix of b.
2. Let k be the first positions where a and b differ. The ASCII code of the character at position k in the string a is smaller than ASCII code of the character at position k in the string b.

Hint:

32		48	0	64	@	80	P	96	‘	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	’	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o		