Problem	DECODE	SHELVES	CALC
Executable file	DECODE.EXE	SHELVES.EXE	CALC.EXE
Source file	DECODE.BAS DECODE.PAS DECODE.C DECODE.CPP	SHELVES.BAS SHELVES.PAS SHELVES.C SHELVES.CPP	CALC.BAS CALC.PAS CALC.C CALC.CPP
Input file	DECODE.IN	SHELVES.IN	CALC.IN
Output file	DECODE.OUT	SHELVES.OUT	CALC.OUT
Time limit per test	10 seconds	10 seconds	10 seconds
Number of tests	10	10	10
Points per test	3	5	7
Total points	**30**	**50**	**70**
	150		

DECODE

A text can be coded so that its letters are replaced by other letters. To do it we use a replacement table that can be created thus:

A word with different letters of English alphabet is chosen – a **key word**. An integer K less than or equal to 26 is chosen – a **key number**. A replacement table has two rows and 26 columns.

The upper row contains sorted letters of English alphabet (all of them). The key word is written letter by letter left to right in lower row starting from K^{th} position. Upon writing the last letter of the key word, we continue to write other letters (not appearing in the key word) sorted lexicographically. When a letter is written in the last (26^{th}) position of lower row, remaining letters are then written starting from the first position.

For example, if the key word is DUBROVNIK and the key number is 10, then the replacement table can be written as

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
M	P	Q	S	T	W	X	Y	Z	D	U	B	R	O	V	N	I	K	A	C	E	F	G	H	J	L

An original text can be coded so that each letter is found in the first row and then replaced by a letter written below it.

Write a program that will using given key word and key number decode given coded text, i.e. find the original text.

Input data

The first line of input file contains a key word consisting of capital letters of English alphabet (A–Z). Length of a key word will be less than or equal to 26.

The second line of input file contains an integer K, $1 \leq K \leq 26$, a key number.

The third line of input file contains coded text consisting of capital letters of English alphabet (A–Z). Length of coded text will be less than or equal to 100.

Output data

The first and only line of output file should contain decoded, i.e. original text.

Examples

DECODE.IN

```
NOVI
15
DTZNMNXAWT
```

DECODE.OUT

```
VINODOLSKI
```

DECODE.IN

```
DUBROVNIK
10
SVPKVSVABZOMSRZY
```

DECODE.OUT

```
DOBRODOSLINADMIH
```

DECODE.IN

```
ZAGREB
23
QYELREDEWEMLFNEIEP
```

DECODE.OUT

```
OVAJZADATAKJELAGAN
```

SHELVES

A rack in a police department consists of shelves that are organised in C columns and R rows.

To reach an object from any shelf, a ladder must be used. A ladder can be leaned against one column of shelves only. By climbing the ladder to a certain height (row) it is possible to take any object from that column up to climbed height including any object from adjacent (left and right) columns.

Policemen need to take certain objects from the rack. To lower a risk of injury at work, it is necessary to find a way to take all needed objects from the rack so that total height of all climbing is minimal possible. Total height is a sum of heights of all climbings.

A rack with some object lying on its shelves is given.

Write a program that will determine minimal possible total height of climbings needed to collect all the objects from the rack.

Input data

The first line of input file contains two integers C and R, $1 \leq C \leq 100$, $1 \leq R \leq 100$, number of columns and number of rows, separated by a space character.

The second line of input file contains an integer N, $1 \leq N \leq 100$, number of shelves that needs to be reached.

Every of next N lines contains two integers A and B, $1 \leq A \leq C$, $1 \leq B \leq R$, separated by a space character, number of column and a row of a shelf that need to be reached.

Output data

The first and only line of output file should contain minimal possible total height of climbings needed to reach all given shelves.

Examples

SHELVES.IN

```
5 5
3
2 3
3 4
4 4
```

SHELVES.OUT

```
4
```

SHELVES.IN

```
6 20
4
5 6
1 1
6 1
3 7
```

SHELVES.OUT

```
9
```

SHELVES.IN

```
10 10
5
9 1
7 6
5 8
4 1
3 2
```

SHELVES.OUT

```
11
```

Hal has a symbolic calculator whose memory is organised as a list that can have at most 100 elements. Each element can contain an arithmetic expression. The calculator understands following commands:

HASH – If the first element of a list contains expression **s** and the second element contain expression **t** then this command removes those two elements from list and creates new first element containing expression **(t#s)**.

DOLLAR – If the first element of a list contains expression **s** and the second element contain expression **t** then this command removes those two elements from list and creates new first element containing expression **(t$s)**.

SWAP – Exchanges the first and the second element of a list.

DROP – Removes the first element from a list.

DUP – New first element is created and expression from the second element of list (which was the first before the creation of new element) is copied to it.

ROT – Rotates the first four elements of list so that the first becomes the second, the second becomes the third, the third becomes the fourth and the fourth becomes the first.

List with two elements is given. The first element contains **A** and the second element contains **B**.

Write a program which will for given expression **e** generate a sequence of commands for Hal's calculator that will, starting with given list and executing all commands of the sequence without any error as a result leave a list whose first and only element contain expression **e**.

An error occurs if list gets too long (attempt to add 101st element) or if there are not enough elements needed to perform a command. Commands DUP and DROP require that the first element has an expression; commands HASH, DOLLAR and SWAP require that the first two elements have expressions; command ROT requires that the first four elements of a list have expressions.

Program should generate **any** sequence of commands that as result leaves given expression. Number of commands in a sequence should not exceed 10000.

Input data

The first and only line of input file contains a given expression which is a sequence of characters **A**, **B**, **#**, **$**, **(,)** and nothing else (not even space characters).

Length of expression will always be 100 or less.

There will always be a solution to test data.

Output data

The output file should contain a sequence of commands generating given expression. Each line should contain one command.

Note: a solution needs not to be unique.

Examples

CALC.IN

```
(B$A)
```

CALC.OUT

```
SWAP
DOLLAR
```

CALC.IN

```
(A$(A#B))
```

CALC.OUT

```
SWAP
DUP
DUP
ROT
HASH
DOLLAR
SWAP
DROP
```

CALC.IN

```
((A$B)#(B$A))
```

CALC.OUT

```
SWAP
DUP
DUP
ROT
ROT
DROP
DUP
DUP
ROT
ROT
DROP
SWAP
ROT
ROT
DOLLAR
DUP
ROT
ROT
ROT
DROP
SWAP
DOLLAR
HASH
```

Explanation of solution to example #2

	SWAP	**DUP**	**DUP**	**ROT**	**HASH**	**DOLLAR**	**SWAP**	**DROP**
2: A	2: B	3: B	4: B	4: A	3: A	2: A	2: (A$(A#B))	1: (A$(A#B))
1: B	1: A	2: A	3: A	3: A	2: A	1: (A$(A#B))	1: A	
		1: A	2: A	2: A	1: (A#B)			
			1: A	1: B				