TASK	ŠEĆER	KOLO	GITARA	POŠTAR	KUGLICE	UPIT
source code	`secer.pas` `secer.c` `secer.cpp`	`kolo.pas` `kolo.c` `kolo.cpp`	`gitara.pas` `gitara.c` `gitara.cpp`	`postar.pas` `postar.c` `postar.cpp`	`kuglice.pas` `kuglice.c` `kuglice.cpp`	`upit.pas` `upit.c` `upit.cpp`
input	standard input (*stdin*)					
output	standard output (*stdout*)					
time limit	1 second	1 second	1 second	4 seconds	1 second	1 second
memory limit	32 MB	32 MB	32 MB	32 MB	32 MB	128 MB
point value	**30**	**50**	**70**	**100**	**120**	**130**
	500					

Mirko works in a sugar factory as a delivery boy. He has just received an order: he has to deliver **exactly N** kilograms of sugar to a candy store on the Adriatic coast. Mirko can use two types of packages, the ones that contain **3 kilograms**, and the ones with **5 kilograms** of sugar.

Mirko would like to take as few packages as possible. For example, if he has to deliver 18 kilograms of sugar, he could use six 3-kilogram packages. But, it would be better to use three 5-kilogram packages, and one 3-kilogram package, resulting in the total of four packages.

Help Mirko by finding the minimum number of packages required to transport **exactly N** kilograms of sugar.

INPUT

The first and only line of input contains one integer **N** ($3 \le \mathbf{N} \le 5000$).

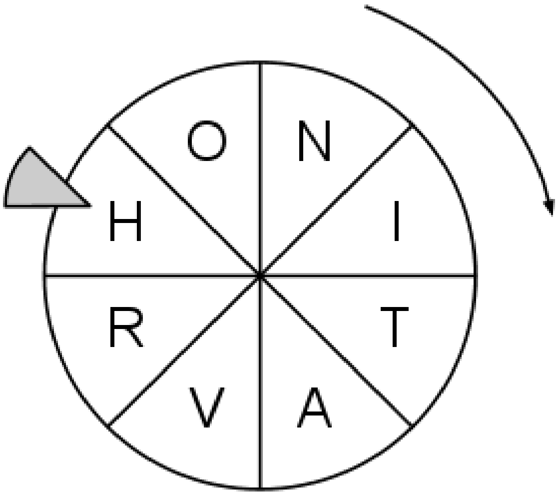
OUTPUT

The first and only line of output should contain the minimum number of packages Mirko has to use. If it is impossible to deliver exactly **N** kilograms, output -1.

SAMPLE TESTS

input 4 **output** -1	**input** 9 **output** 3	**input** 18 **output** 4

Mirko has recently bought a wheel of fortune. He wrote an uppercase letter of English alphabet onto each wedge, like this (3rd example test case):



No letter appears twice in the wheel, and the wheel spins in clockwise direction. There is a pointer that stays in the same place while the wheel is spinning (it is pointing to H in the picture above). When we spin the wheel, the letter to which the pointer is pointing to changes accordingly.

Mirko spinned the wheel **K times in a row**, and each time he wrote down how many times the pointed letter changed, and what letter was pointed to at the end of that spin.

Slavko found that paper, and would like to now what letters Mirko wrote onto the wedges of the wheel. Help him determine this, if the total number of wedges is known.

INPUT

The first line of input contains integers **N** (2 ≤ **N** ≤ 25), the number of wedges on the wheel, and **K** (1 ≤ **K** ≤ 100), the number of spins.

The following **K** lines contain descriptions Mirko wrote down for each spin, in order. Each line contains an integer **S** (1 ≤ **S** ≤ 100), the number of times the pointed letter changed during that spin, and an uppercase letter at which pointer stopped.

OUTPUT

If there is no wheel that meets the requirements described, output '!'.

Otherwise, output sequence of letters written onto the wheel, starting from the pointed letter at the end of the last spin and proceeding clockwise. If some letter can't be determined, output '?' instead.

SAMPLE TESTS

input 3 3 1 A 2 B 3 C **output** !	**input** 5 6 1 A 2 B 5 B 1 C 2 A 2 B **output** B?A?C	**input** 8 8 4 V 3 I 7 T 7 A 6 R 5 N 1 O 9 H **output** HONITAVR

Darko has a new imaginary extraterrestrial friend with a **billion** of fingers. The extraterrestrial has soon took hold of his guitar, found a simple melody online, and started playing it.

The guitar, as usual, has **six** strings denoted by numbers 1 through 6. Each string is divided into **P frets** denoted by numbers 1 through **P**.

A melody is a sequence of tones, where each tone is produced by picking a string pressed on a specific fret (e.g. 4th string pressed on the 8th fret). If a string is pressed on several frets, the produced tone will be the one corresponding to the **highest** of those frets.

For instance, if the 3rd string is already pressed on the 5th fret, and the tone which corresponds to the 7th fret is to be produced, the string can be pressed on the 7th fret and picked without releasing the 5th fret, since only the highest one affects the tone produced (7th in this case). Similarly, if a tone that corresponds to the 2nd fret on the same string is next to be produced, it is necessary to release both 5th and 7th frets.

Write a program which computes the minimum number of finger movements needed to produce the given melody. Note that press or release a single fret counts as one finger move. String picking does not count as finger move, but rather a guitar pick move.

INPUT

The first line of input contains two positive integers separated by a single space, **N** (**N** ≤ 500 000) and **P** (2 ≤ **P** ≤ 300 000). They represent the number of tones in the melody and the number of frets, respectively.

The following **N** lines describe the fields for the corresponding tones – the ordinal of the string and the ordinal of the fret, respectively, in the same order as the extraterrestrial play them.

OUTPUT

In a single line of output, print the minimum number of finger movements.

SAMPLE TESTS

input	input
5 15 2 8 2 10 2 12 2 10 2 5	7 15 1 5 2 3 2 5 2 7 2 4 1 5 1 3
output	**output**
7	9

First sample description: all the tones played are produced by picking the 2nd string. First, the frets 8, 10, 12 are pressed, in order (three movements). Then, the 12th fret is released to produce the second tone again (fourth movement). Finally, the 5th fret is pressed followed by the release of frets 10 and 12 (a total of seven movements).

Second sample description: 1, 1, 1, 1, 3, 0, 2 finger movements are necessary, in the order of the seven tones produced.

Mirko has got a mailman job in a small town in the hills. The town can be represented by a **N**×**N** matrix. Each field contains one of the following, exclusively: a house denoted by 'K', the post office denoted by 'P', or a pasture denoted by '.'. Additionally, each field is assigned an altitude.

Every morning, Mirko delivers mail to all houses in the town. He starts at the field denoted by 'P', which represents a single post office in the town. Mirko is allowed to move horizontally, vertically and diagonally, to adjacent squares only. Once he delivers the last piece of mail, he must return to the post office.

Mirko did not have a clue about how tiresome his job will be. Let the difference between the heights of the highest and the lowest field Mirko visits while delivering the mail be equal to his tiredness. Help him out and determine the least tiredness possible for Mirko to deliver all the mail.

INPUT

The first line of input contains an integer **N** (2 ≤ **N** ≤ 50).

The following **N** lines represent fields in the corresponding matrix row. The character 'P' will appear **exactly once**, while the character 'K' will appear **at least once**.

The following **N** lines each contain **N** positive integers, the altitudes of the fields in the corresponding matrix row. Those values are less than 1 000 000.

OUTPUT

In a single line of output print a single integer that represents the minimum possible tiredness.

SAMPLE TESTS

input 2 P. .K 2 1 3 2 **output** 0	**input** 3 P.. .KK ... 3 2 4 7 4 2 2 3 1 **output** 2	**input** 3 K.P ... K.K 3 3 4 9 5 9 8 3 7 **output** 5

First sample description: Starting from the post office, Mirko can move directly to the field with the house, deliver the mail and return back to the post office. Since both the field with the post office and the one with the house have the same altitude, Mirko's tiredness is equal to zero.

Mirko and Slavko love playing with marbles. On an exciting Friday, Mirko has come up with a marble game which he wants to show to Slavko.

In the game, Mirko constructs a **directional** graph in which all vertices have at most **at most one** outgoing edge. Then he places a marble on one of the vertices. Whenever a marble is on some vertex X, the marble moves to the adjacent vertex connected by a single edge, if such exists. The movement of the marble continues until a vertex with no outgoing edge is visited, where the marble stops. It is also possible that the marble traverses the graph indefinitely in case no such vertex is ever visited.

To make sure that Slavko understand the rules of the game, Mirko will ask a series of queries. The types of queries are as follows:

1 X – unless the marble stucks in a loop, on which vertex will the marble stop if it is placed on the vertex X?

2 X – delete the outgoing edge of the vertex X (it is guaranteed that such edge will always exist)

Note: queries are executed in order and are cummulative (one affects another).

INPUT

The first line contains a postive integer **N** ($1 \le$ **N** $\le 300\ 000$), the number of vertices in the graph.

The second line contains exactly **N** positive integers separated by a single space, where the number at position **i** denotes the index of the destination of the outgoing edge from vertex with index **i**. The zero value represent that there is no outgoing edge from the vertex with index **i**.

The following line contains a single positive integer **Q** ($1 \le$ **Q** $\le 300\ 000$), the number of queries.

The remaining **Q** lines contain queries in the format described above.

OUTPUT

For each query of type 1, output the index of the vertex where the marble stops, one per line, in the order of the execution of queries. If the marble never stops, output **CIKLUS** instead.

SAMPLE TESTS

input 3 2 3 1 7 1 1 1 2 2 1 1 2 1 1 2 2 1 2 **output** CIKLUS CIKLUS 1 1 2	**input** 5 0 3 5 3 4 6 1 1 1 2 2 4 1 2 2 3 1 2 **output** 1 CIKLUS 4 3

Mirko got tired of implementing all kinds of data structures for different tasks. So, he decided to come up with the ultimate structure, one that will allow him to manipulate with his favorite number sequence. Help him!

Mirko will give you his number sequence, and a sequence of queries you must execute. Each query either asks for information, or modifies the existing sequence. Possible query types are listed below.

Query type	**Description**	**Example**
1 A B X	Set all elements from Ath to Bth (inclusive) to value X	(9, 8, 7, 6, 5, 4, 3, 2, 1) → *1 3 5 0* → (9, 8, **0, 0, 0**, 4, 3, 2, 1)
2 A B X	Add X to Ath element, 2*X to (A+1)th, …, and (B-A+1)*X to the Bth element	(9, 8, 7, 6, 5, 4, 3, 2, 1) → *2 3 5 2* → (9, 8, **9, 10, 11,** 4, 3, 2, 1)
3 C X	Insert new element with value X immediately before the Cth element	(9, 8, 7, 6, 5, 4, 3, 2, 1) → 3 4 100 → (9, 8, 7, **100**, 6, 5, 4, 3, 2, 1)
4 A B	Find the sum of all elements from Ath to Bth	(2, 18, 7, 6, 1, **4, 7**, 7, 2) → 4 6 7 → result: 11

INPUT

The first line of input contains integers **N** and **Q** ($1 \le$ **N**, **Q** $\le 100\ 000$), the starting sequence length and the number of queries.

The following line contains the starting sequence. Sequence consists of non-negative integers not greater than 100000 that are separated by a single space.

The following **Q** lines contain queries in the format described above. In all queries, $1 \le X \le 100$, $1 \le A \le B \le currentSequenceLength$, and $1 \le C \le currentSequenceLength+1$.

OUTPUT

For each query of type 4 output one line containing the requested sum.

Note: notice that some sums won't fit into 32-bit integer data type.

SAMPLE TESTS

<table>
<tr>
<td>

input

```
5 5
1 2 3 4 5
1 5 5 0
4 4 5
4 5 5
2 1 5 1
4 1 5
```

output

```
4
0
25
```

</td>
<td>

input

```
1 7
100
3 1 17
3 2 27
3 4 37
4 1 1
4 2 2
4 3 3
4 4 4
```

output

```
17
27
100
37
```

</td>
</tr>
</table>