An Archeological Puzzle

The year is 2242. Humanity has mastered deep space travel, but our greatest discoveries lie buried on our own planet. You're an archaeolinguist specializing in the ancient digital civilizations of the 21st century.

During a recent dig, your team unearthed a data cube from what's believed to be a pre-cataclysmic military installation. The cube contains a series of encrypted messages, but with a twist—the encryption key is a set of instructions, not a traditional password. These instructions, written on a laminated post-it found next to the cube, describe a process to revert the scrambled data back to its original state. Your task is to write a program that can follow these instructions to decipher the messages.

The Decryption Protocol

The instructions on the post-it are composed of two distinct characters: R and D.

- The character R stands for "Reorient," which reverses the numerical sequence of the message.
- The character D stands for "Disassemble," which removes the first element of the sequence.

The 21st century civilization's technology was prone to data corruption. If the protocol ever attempts to "Disassemble" an empty sequence, it means the data is irrevocably corrupted. Your program must be able to detect this and report "error" instead of trying to recover the message. Reorienting an empty sequence is not an issue, as it simply returns an empty sequence.

Input

The first line contains the decoding instructions from the post-it consisting of characters 'R' and 'D'. The length of the instruction sequence is between 1 and 10^5 (inclusive).

The next line contains the number of values in the encrypted message, n ($0 <= n <= 10^5$).

The last line contains the encrypted message itself $[y_1, ..., y_n]$, where ($1 \leq y_i \leq 100$). The values are enclosed in a set of square brackets (i.e., start with '[' and end with ']'). The individual values are separated by commas and nothing else.

Output

A single line containing either the decrypted sequence or "error". The resulting list should start with '[' and end with ']'. The integers in the input and output list are separated by commas and nothing else.

Example 1

```
Input:
DD
1
[42]

Output:
error
```

Example 2

```
Input:
RDD
4
[1,2,3,4]

Output:
[2,1]
```

Example 3

```
Input:
RRRDD
4
[1,2,3,4]

Output:
[2,1]
```

Example 4

```
Input:
RRRDDRRDD
4
[1,2,3,4]

Output:
[]
```

Note: there is no output from this sequence since all values are removed. The program should print an empty list.

Stack Puzzle

There are two sequences of stack operations converting the word TROT to TORT:

```
[
i i i i o o o o
i o i i o o i o
]
```

where `i` and `o` stands for *in* / push and *out* / pop operation respectively. In this problem, you are given two words and you are asked to find out all sequences of stack operations converting the first word to the second word.

Input The input consists of two lines, the first of which is the source word and the second is the target word. The length of each word does not exceed 12.

Output Your program should print a sorted list of valid `i/o` sequences. The list is delimited by

```
[
]
```

and the sequences are sorted in lexicographical order. Within each sequence, `i`'s' and `o`'s are separated by a single space and each sequence is terminated by a new line. There should be a newline character printed after the closing square bracket delimiter. Warning: there should not be any spaces after the last `o` of the output.

Process Given an input word, a valid `i/o` sequence implies that every character of the word is pushed and popped exactly once, and no attempt is ever made to pop an empty stack. For example, if the word FOO is input, then the sequence: + `i i o i o o` is valid and produces OOF + `i i o` is not valid (too short), + `i i o o o i` is not valid (illegal pop from an empty stack)

A valid sequence results in a permutation of the letters in the input word. For example, given the input word FOO, both sequences `i i o i o o` and `i i i o o o` give the word OOF.

Example 1

```
Input:
madam
adamm

Output:
[
i i i i o o o i o o
i i i i o o o o i o
i i o i o i o i o o
i i o i o i o o i o
]
```

Example 2

```
Input:
bahama
bahama

Output:
[
i o i i i o o i i o o o
i o i i i o o o i o i o
i o i o i o i i i o o o
i o i o i o i o i o i o
]
```

Example 3

```
Input:
long
```

```
short

Output:
[
]
```

Example 4

```
Input:
eric
rice

Output:
[
i i o i o i o o
]
```

Job Scheduling

At the supercomputing facility researchers request computing time in one of two different ways:

- one time tasks that have a fixed start time and end time, for example a task with start time 110 and end time 150 needs to run during the time [110..150]
- repeating tasks that need to run for ever in fixed time intervals, for example a task with start time 25, end time 38 and repeating interval of 100 needs to run at times [25..38], [125..138],[225..238],...

Because the research simulations they are running are highly time sensitive, only one task can execute at any given time.

Your task as a new intern in the facility is to make sure that no computing time requests conflict in their time slots.

You are given N one-time tasks and M repeating tasks and you need to determine if there is any overlap between them in the time interval [0..1,000,000].

Note: tasks are considered to overlap only if their time intervals overlap, but not if the endpoints are the same. For example, [2..5] and [4..6] are overlapping, while [2..4] and [4..6] are not.

Input

The first line of the input contains two integers N and M (0 <= N, M <= 100), indicating the number of one-time tasks and repeating tasks, respectively. Each of the following N lines contains two integers indicating the start time and the end time for the one-time tasks. Afterward, each of the following M lines contains three integers indicating the start time, end time and repetition interval of the repeating tasks.

It is guaranteed that all integers are between 0 and 1,000,000, the end time is larger than the start time and the repetition interval is positive.

Output

Print one line either containing `NO CONFLICT` if there is no overlap, or `CONFLICT` if there are any overlaps.

Example 1

```
Input:
2 1
5 10
10 15
1 2 20

Output:
NO CONFLICT
```

Example 2

```
Input:
0 3
1 5 20
105 106 100
90 105 500

Output:
CONFLICT
```

Example 3

```
Input:
3 1
2 5
5 6
6 8
10 15 100
```

Output:
NO CONFLICT

Prime Gap

A **prime gap** is the difference between two successive prime numbers. The n-th prime gap, denoted g(n) is the difference between the (n+1)-th and the n-th prime numbers, i.e.

g(n) = p(n+1) - p(n)

The first 7 prime numbers are 2, 3, 5, 7, 11, 13, 17, and the first 6 prime gaps are 1, 2, 2, 4, 2, 4.

Shinya Yukimura is interested in prime gaps and he needs some experimental data to verify his hypothesis. More specifically, given a closed interval [a,b], Shinya wants to find the two adjacent primes p1 and p2 (a <= p1 <= p2 <= b) such that the prime gap between p1 and p2 is minimized (i.e. p2-p1 is the minimum). If there are multiple prime pairs that have the same prime gap, report the first pair. Shinya also wants to find the two adjacent primes p3 and p4 (a <= p3 < p4 <= b) that maximize the gap between p3 and p4 (choose the first pair if there are more than one such pairs).

Please write a program to help Shinya.

Input

Two integer values a,b, with a < b. The difference between a and b will not exceed 1,000,000. 1 <= a <= b <= 2,147,483,647.

Output

If there are no adjacent primes in the interval [a,b], output -1 followed by a newline.

Otherwise the output should be 4 integers: `p1 p2 p3 p4` as mentioned above separated by a space and followed by a newline (no space after p4).

Example 1

```
Input:
1 20

Output:
2 3 7 11
```

Example 2

```
Input:
13 16

Output:
-1
```

In the first example test case, the prime gap between 13 and 17 also has the largest value 4, but the pair (7,11) appears before (13,17), so we output 7 11 instead of 13 17.

Chasm Crossing

Imagine you're managing a unique transportation system that links two sides of a city separated by a vast chasm. This system uses a single aerial tramway with a maximum capacity of N pods, and each one-way journey across the chasm takes T minutes.

Citizens arrive at either the 'Central' station or the 'Uptown' station to catch the tram. The tram operates continuously as long as there's at least one passenger or an empty pod waiting at either station. Upon arrival, the tram unloads all passengers and then loads up to N waiting passengers on a first-come, first-served basis. If there are no passengers waiting at either station, the tram waits until someone arrives. Initially, the tram is positioned at the 'Central' station.

Your task is to determine the precise time each passenger arrives at the opposite station.

Input

The first line of input provides three integers: N, the number of pods; T, the one-way travel time in minutes; and M, the total number of passengers. The next M lines each contain a passenger's arrival time and their departure station ('Central' or 'Uptown'). Passengers are listed in the order they arrive (their arrival times are non-decreasing), and the time spent on loading and unloading is negligible.

The first line of input contains three integers `N`, `T` and `M` (`1 <= N, T, M <= 10,000`). Each of the following `M` lines gives the arrival time of a car and the bank at which the car arrives (`left` or `right`). The cars are ordered by their arrival times (so the arrival times are non-decreasing) and the time spent on loading and unloading can be ignored.

Output For each passenger, you need to print a single line showing the time they are unloaded at their destination station.

WARNING: the order of passengers in the output should match the order in the input.

Example 1

```
Input:
2 10 10
0 Central
10 Central
20 Central
30 Central
40 Central
50 Central
60 Central
70 Central
80 Central
90 Central

Output:
10
30
30
50
50
70
70
90
90
110
```

Figure 1: Illustration generated by Gemini.

Example 2

```
Input:
2 10 3
10 Uptown
```

```
25 Central
40 Central

Output:
30
40
60
```