Problem name: Srba

Language: English

Source code: **srba.c, srba.cpp, srba.pas**
Input / Output files: **standard input / output**

Task description

Srba has a wonderful plum orchard in the heart of Šumadija. He likes to keep things neat, so he planted all of the plum trees in a single row, and he keeps track of the heights of each tree.

When Srba climbs a plum tree, he notices that his row is not perfectly straight so he can fully see the tops of some other trees. Actually, he can fully see all other trees which are **strictly shorter** than the tree he is on.

This has made him curious, so he now wants to know what is the furthest away tree he can see from the top of each plum.

Input

The first line of input contains one integer, *N* – the number of plum trees. The second line contains N integers $H_i, 1 \leq i \leq N$ – heights of each plum tree in order.

Output

The first and only row of output should contain N integers. For each i between 1 and N, i-th of these numbers is the index (starting from 1) of the furthest away shorter tree when observed from tree i. If no trees satisfy the condition, the i-th number should be -1. In case of a tie between multiple trees, the one with the **smallest index** should be written to the output.

Example

Input:	Output:
7 2 9 3 5 1 1 4	6 7 6 1 -1 -1 1

Example explanation

For the first tree, the two shorter trees are at indices 5 and 6, and the one with index 6 is further away from it. The second tree is the tallest one, and the one furthest away with it is at the right end – index 7. For the other five trees the process is very similar.

Constraints

- $2 \leq N \leq 10^6$
- $1 \leq H_i \leq 10^9$
- In 40% of test cases it holds $N \leq 5000$.

Problem name: Marathon

Language: English

Source code: **marathon.c, marathon.cpp, marathon.pas**
Input / Output files: **standard input / output**

Task description

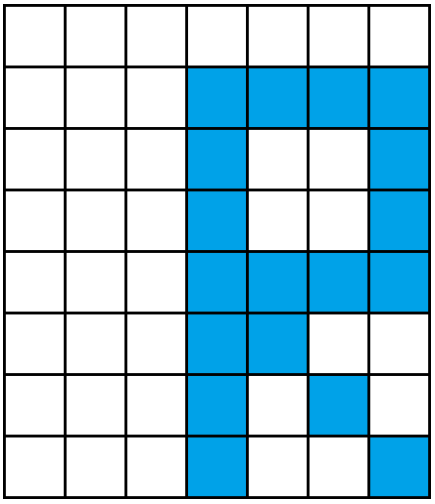
Once a year a famous marathon is organized at mountain Fruska Gora in Serbia. This year competitors complained that there are too many up-hills and down-hills on the route, so for the next year organizers want to pick such a route that a whole route is at the same altitude. Additionally to that, mascot of the next year marathon is Rabbit, so track has to be in the shape of letter 'R'.

We can represent mountain as a matrix with N rows and M columns where element at i-th row and j-th column represent altitude of the mountain at that point(i, j). Altitudes are represented with colors and each color is represented with one lowercase letter 'a' to 'z'. Fields with same altitude are represented with same color.

We say that a track that begins at position (r, c) has the shape of letter 'R' if it has following route:

1. $2k$ fields north – from point (r, c) to point $(r - 2k, c)$ in matrix
2. k fields east – from point $(r - 2k, c)$ to point $(r - 2k, c + k)$ in matrix
3. k fields south – from point $(r - 2k, c + k)$ to point $(r - k, c + k)$ in matrix
4. k fields west – from point $(r - k, c + k)$ to point $(r - k, c)$ in matrix
5. k fields south-east – from point $(r - k, c)$ to point $(r, c + k)$

One possible route, starting in (8, 4) with k = 3, is shown in the figure 1.



- Figure 1 -

Organizers are interested to find out how many routes that has the shape of letter $'R'$ are there, such that all fields on route have the same altitude.

Input

In the first line of standard input are 3 positive integers N, M and K. In the next N lines there are M lowercase characters, where j-th character in row $i+1$ represent the altitude at point (i, j).

Output

In the first and only line of standard output you have to write the number of routes that has the shape of letter $'R'$ such that all points on that route have the same altitude.

Example

Input:	Output:
8 7 3 aaacccc azabbbb aaabbcb aazbaab ayabbbb cccbbtt cccbcbp cccbeeb	1

Example explanation

Figure 1 shows the only possible route.

Constraints

- $1 \leq N, M, K \leq 1000$
- In 40% of test cases it holds $1 \leq N, M, K \leq 100$

Problem name: Santa

Language: English

Source code: **santa.c, santa.cpp, santa.pas**
Input / Output files: **standard input / output**

Task description

At the North Pole, Santa Claus and his assistants, elves, have a lot of work this year. Presents are being made on two tapes (conveniently numbered tape 1 and tape 2) and elves stand alongside them. When an elf comes to work, Santa sends him to the start of one tape (the other elves on that tape move in order to make space for the new elf). Elves have a habit (for unknown reason) to change their tape from time to time, and therefore the elf at the beginning of one tape can go to the beginning of the other tape.

Because all elves are equal, they leave work in the same order in which they came (so the first elf that came to work has to be the first one to leave, etc.). But, in order to not mess up the working process, the elf can leave the work only if it is at the beginning of one of the tapes.

Your job is to move the elves from one tape to the other in order to achieve that at the moment the next elf is about to leave the work, he is at the beginning of one of the tapes. Your program needs to answer the following two types of queries:

'1': (a new elf comes to work) – your program needs to print the number of the tape this elf will go to

'2': (the next elf leaves work) – your program needs to print the number of the tape where the elf that is about to leave is located

At any time, you are allowed to issue any of the swap commands, '1 2' or '2 1'. In the former case the elf at the beginning of tape 1 will be moved to the beginning of tape 2, and in the latter case the elf at the beginning of tape 2 will be moved to the beginning of tape 1.

Input

The first line of the input contains one integer N that represents the number of elves. In the next line there is a string consisting of 2N characters ('1' and '2'), where each character represents one query. The input string will be valid, i.e. there will be N queries of type '1', and N queries of type '2', and at any moment the number of queries of type '1' will be at least as big as the number of queries of type '2'.

Output

Each row of output should contain either an answer to a query or a swap command. Your program should answer the queries in the same order as in the input. The swap commands can be issued at any point and the number of them is not limited. However, the chronological order of commands has to

be correct, e.g. if a swap needs to occur before an elf can exit a tape, in the output the swap command has to be above the answer to that query.

Example

Input:	Output:
4 11122122	1 2 1 1 2 1 2 1 2 2 1 2

Example explanation

The first elf comes to the work and we put it at the beginning of the tape 1, we put the 2nd elf at the beginning of the tape 2, and the 3rd elf at the beginning of the tape 1.

When the 1st elf is about to leave, we move the 3rd elf from the beginning of tape 1 to the beginning of tape 2, and therefore the 1st elf is at the beginning of tape 1, so we return 1. When the 2nd elf is about to leave, we move the 3rd elf from the beginning of the tape 2 to the beginning of the tape 1, and since the 2nd elf is then at the beginning of the tape 2, we return 2.

When the 4th elf comes, we put it at the beginning of the tape 2, and at that moment we have 3rd elf at the beginning of tape 1 and 4th elf at the beginning of the tape 2, and once the 3rd elf is about to leave, we print 1, and when the 4th elf is about to leave we print 2.

Constraints

- $2 \leq N \leq 100000$
- In 50% of test cases it holds $N \leq 1000$

Problem name: Jam

Language: English

Source code: **jam.c, jam.cpp, jam.pas**
Input / Output files: **standard input / output**

Task description

Belgrade, the city host of the 1st Mathematical Grammar School Cup, is a great place for sightseeing. It consists of the N city squares (labeled with numbers from 1 to N) and M **one-way** streets (labeled with numbers from 1 to M) between some of them. The configuration of the city is such that **it is impossible to start at some city square and, using the streets, end at that same city square again** (i.e. there are no "street-cycles"). **We are located at city square 1** and we plan to visit some other city squares.

Unfortunately, during the day, in some of the streets traffic jam occurs and those streets become blocked, i.e. we cannot use them anymore. Your job is to determine, given the map of the city and information about traffic jams, is it possible to reach some given destination.

More precisely, Q events occur **in the given order**. Each event is one of the following:

- $1\ x$ – Traffic jam occurs at the street number x. This street is blocked and remains blocked during the following events.
- $2\ y$ – We want to go from city square 1 to city square y via some **non-blocked** streets. **You need to determine if this is possible.**

Note that some city square a is reachable from the city square 1 via some non-blocked streets if there exists some city squares $s_1, s_2, \ldots, s_k$ such that there is a non-blocked street from s_i to s_{i+1} for all $1 \leq i < k$ and $s_1 = 1, s_k = a$ (i.e. there is a path $1 \to s_2 \to \cdots \to s_{k-1} \to a$). **At the beginning, none of the streets are blocked and all city squares are reachable from the city square 1.**

Input

The first line of input contains two integers, N and M – the number of city squares and the number of one-way streets, respectively. Next M lines contain two integers a_i and b_i, separated with one space, denoting the one-way street from city square a_i to city square b_i. Next line contains one integer Q – number of events. Each of the following Q lines contain two integers separated with one space - representing the event, as described above. **Note that the streets are numbered in order given in the input.**

Output

For each event of the form "$2\ y$", you need to write number 1 if it is possible to reach city square y via non-blocked streets or number 0 otherwise. Each number should be written in a new line and the order must be the same as the order of the questions in the input.

Example

Input:	Output:
4 5 1 4 2 4 2 3 1 2 4 3 6 2 2 2 3 1 4 2 2 1 1 2 3	1 1 0 0

Example explanation

We have 4 city squares and 5 one-way streets: $1 \to 4, 2 \to 4, 2 \to 3, 1 \to 2$ and $4 \to 3$. First event is a question: can we reach city square 2 via non-blocked street? The answer is $'1'$ since there is a non-blocked street $1 \to 2$. Second event is a question: can we reach city square 3? The answer is again $'1'$ (for example, using path $1 \to 2 \to 3$ or $1 \to 4 \to 3$ or $1 \to 2 \to 4 \to 3$). Next event is to block the street number 4 which is street $1 \to 2$. Now we cannot reach the city square 2 and the answer to the next event is $'0'$. After that we block street number 1 ($1 \to 4$). Now we cannot reach city square 3 from city square 1 and the answer to the last evetn is $'0'$.

Constraints

- $2 \leq N \leq 50.000, 1 \leq M \leq 200.000, 1 \leq Q \leq 100.000.$
- For each street, $a_i \neq b_i$ and $1 \leq a_i, b_i \leq N$.
- Between any two city squares there is at most one one-way street.
- For each event $'1\ x'$ it holds $1 \leq x \leq M$ and for each event $'2\ y'$ it holds $2 \leq y \leq N$.
- Each street will be blocked at most once.
- In 40% of test cases it holds $N, M, Q \leq 1.000$.
- In 20% of test cases it holds $M = N - 1$, and the city map is $1 \to 2 \to 3 \to \cdots \to N$.