Flash memory

1 sec / 10 sec **1 GB**

A microcontroller chip has B bits of built-in flash memory. We need to store and update an M-bit variable in there. Flash memory has a limitation that bits can be changed from `0` to `1` individually, but changing from `1` to `0` is possible only by erasing the whole memory. The memory can be erased only a limited number of times before the chip is worn out and has to be replaced. Therefore, it is desirable to be able to write as many values as possible before erasing.

Your task is to devise an efficient way of storing values to flash memory so that the current value can always be retrieved. More precisely, your program has to implement two operations:

- Writing a value: The inputs for this operation are the current state of the memory and the new value to be written and you have to output the new state of the memory.
- Reading a value: The input for this operation is the state of the memory after some number of write operations and you have to output the value that was written in the last write operation.

Your reading and writing operations can not exchange information between them in any other way besides reading the current state of the memory from input and the writing operation changing some bits (possibly none) from `0` to `1` before writing the new state to output.

Interaction. This is an interactive task. When your program starts, the first line of input contains integer T, where $T = 0$ means your program will write values to memory and $T = 1$ means it will read values from memory. The second line contains the integers B and M. The following lines describe the operations.

For both writing and reading, the first line of each operation contains integer C, where $C = 0$ means there will be no more requests and your program should exit, but $C = 1$ means your program should continue.

- For $T = 0$ and $C = 1$, the second line contains two space-separated strings: the current state of the memory as a sequence of B bits and the new value to be written as a sequence of M bits. If your program can write the new value to the memory by only changing some bits from `0` to `1`, it should first output the integer 1 and on the next line the new state of the memory as a sequence of B bits. If your program can't write the new value to the memory, it should output the integer 0.
- For $T = 1$ and $C = 1$, the second line contains a single string: the state of the memory as a sequence of B bits and your program should output a single string: a sequence of M bits, the value that was last written to the memory.

Example. Input Output

```
0
6 2
1
111111 00
                 0
1
000000 11
                 1
                 110000
0
```

In this example, your program is started to write 2-bit values into 6-bit memory. The first request is to write the value `00`, which your program is not able to do. The second request is to write the value `11`, which your program is able to do. Note that the current state of the memory for the second request does not match your program's output after the first request.

Example. Input / Output

```
Input           Output
1
6 2
1
110000
                11
1
110100
                01
0
```

In this example, your program is started to read 2-bit values from 6-bit memory. The first request is to read from the memory state `110000`, from which your program extracts the value `11`. The second request is to read from the memory state `110100`, from which your program extracts the value `01`.

Remark. To ensure your responses are delivered to the grading environment, you have to flush the output buffer after each response (also note that the output always ends with a newline):

Language	Command
C	`fprintf(stdout, "1\n%s\n", s);` `fflush(stdout);`
C++	`cout << 1 << "\n" << s << endl;`
Java	`System.out.println("1");` `System.out.println(s);` `System.out.flush();`
Python	`sys.stdout.write("1\n{0}\n".format(s))` `sys.stdout.flush()`

Testing. For each test case, 4 instances of your program will be started simultaneously, 2 for writing and 2 for reading. The memory and CPU time limits are for all these instances combined. **Any deliberate attempt to pass data out-of-band between these instances is considered cheating and will be cause for disqualification.**

Several blocks of memory, each B bits, will be initialized with zeros. Then writing and reading operations will be performed in some feasible order.

During a writing operation, a writer instance is given the current state of one of the blocks and a value to write into it. You may assume the values to be written have been chosen uniformly randomly from the range $0 \ldots 2^M - 1$ and independently of everything else. If your program is able to write the value, the state of the block will be set to the one returned by your program. If your program can't write the value, that block won't be part of any further writing operations.

During a reading operation, a reader instance is given the state of a block after a successful writing operation. It is checked whether the value returned by your program is the same as what was supposed to be written during the write operation. The read operation will be performed exactly once on the output of each successful write operation.

Grading. In each test group, your program's score is proportional to the average number of values written in tests within this group. More precisely, if your program can write V values per block on average, its score will be $100 \cdot V/P\%$ of the value of the test group, where P is as given below. When your program returns a wrong value in any reading operation, the score for the whole group will be zero. For any other failure, the number of values read in that test will be counted as zero.

The test groups satisfy the following conditions:

1. (5 points) $B = 16$, $M = 8$, $P = 4.06244502449506962 4056$.
2. (5 points) $B = 32$, $M = 8$, $P = 12.26490484130096483 4177$.
3. (5 points) $B = 32$, $M = 16$, $P = 4.12959151370778480 2006$.
4. (5 points) $B = 64$, $M = 8$, $P = 30.03927789426882890 0030$.
5. (5 points) $B = 64$, $M = 16$, $P = 12.95314809421736043 2715$.
6. (5 points) $B = 64$, $M = 32$, $P = 4.07355978823366150 1537$.
7. (5 points) $B = 128$, $M = 8$, $P = 69.77789222892874754 8775$.
8. (5 points) $B = 128$, $M = 16$, $P = 34.73179127514363524 0976$.
9. (5 points) $B = 128$, $M = 32$, $P = 13.95078898770563890 8663$.
10. (5 points) $B = 128$, $M = 64$, $P = 4.03991821060480013 3907$.
11. (5 points) $B = 256$, $M = 8$, $P = 174.46804708607103851 1453$.
12. (5 points) $B = 256$, $M = 16$, $P = 82.22261415140417733 4554$.
13. (5 points) $B = 256$, $M = 32$, $P = 37.62938226976920648 8916$.
14. (5 points) $B = 256$, $M = 64$, $P = 14.26346228205414057 7686$.
15. (5 points) $B = 256$, $M = 128$, $P = 4.01556909389394343 0859$.
16. (5 points) $B = 512$, $M = 16$, $P = 204.74624212741034617 0221$.
17. (5 points) $B = 512$, $M = 32$, $P = 91.77859514807311153 9847$.
18. (5 points) $B = 512$, $M = 64$, $P = 39.23027924214593871 2621$.
19. (5 points) $B = 512$, $M = 128$, $P = 15.00000000216767226 8601$.
20. (5 points) $B = 512$, $M = 256$, $P = 4.00542327711105546 8876$.

Additionally, all test cases satisfy $N \cdot B \leq 120\,000$, where N is the maximal number of write operations your program may be asked to perform.

During the contest, your solution will be graded on a small set of tests for each group. After the contest, your last submission and the submission that scored best on the small test set will be tested on a bigger test set and the best of those two will be your final score for this task. **The score you see in CMS during contest is not your final score.** The purpose of this is to increase the accuracy of your score. Grading on full test set during contest would be too slow.

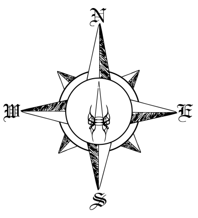
Nautilus

1 sec / 10 sec **256 MB**

Nautilus is a secret submarine, sailing around the ocean and trying to remain hidden.

The ocean is modelled as $R \times C$ grid of cells, where "`#`" represents islands and "`.`" represents water. For example:

```
...##....
..#.##..#
..#....##
.##...#..
....#....
```



Every minute, Nautilus emits a radio signal that can reveal the direction the submarine is about to take. The direction is always one of the following: North (`N`), East (`E`), South (`S`), West (`W`), as shown on the figure above right.

Vytautas has constructed a radar that intercepts the periodic submarine signals. Over the last M minutes, the radar has collected M radio signals, represented as a sequence of M characters, for example "`WS?EE??`". Some of the signals could not be decoded, these are marked as "`?`".

Vytautas does not know the initial submarine location, but wants to use the ocean map in order to identify its current location. Given that Nautilus always stays in water cells on the map, help Vytautas calculate the number of distinct cells where Nautilus may be located at currently.

Input. The first line of the input contains three integers R, C, M.

The next R lines form an $R \times C$ grid of characters "`#`" and "`.`" representing the ocean map.

The last line of the input describes the signals intercepted by Vytautas — a string of M characters, all belonging to the set {`N`, `E`, `S`, `W`, `?`}.

Output. Output a single integer: the number of possible distinct current positions of Nautilus.

Example.

Input	Output

`5 9 7`
`...##....`
`..#.##..#`
`..#....##`
`.##...#..`
`....#....`
`WS?EE??`
	`22`

Grading. The test groups satisfy the following conditions:

1. (29 points) $1 \le R, C, M \le 100$; there are no "`?`" symbols.
2. (37 points) $1 \le R, C, M \le 100$.
3. (34 points) $1 \le R, C \le 500$; $1 \le M \le 5\,000$.

Alpine valley

3 sec / 10 sec **256 MB**

In an Alpine valley there are N villages (numbered $1 \dots N$) connected by only $N - 1$ roads. While it is still possible to get from any village to any other village, this might take quite some time. This gets particularly annoying if you have to buy basic supplies, as there is a shop in only S of the N villages.

This winter the situation got even worse due to heavy snowfall. It would therefore be advisable to either leave the valley, i.e. get to the only village E at the mountain pass connecting the valley to the outside world, or at least buy enough supplies for the next months. You overheard on the radio this morning that the snow has rendered one of the $N - 1$ roads unusable — however, you couldn't clearly understand which one.

You now want to know whether you and your friends can leave the valley and, if not, how far each of you has to drive at least to get to a village with a shop. As you are not sure yet which road is blocked and as your friends live in different villages across the valley, you should write a program that answers this question for Q given combinations of village and blocked road.

Input. The first line contains the integers N, S, Q, and E, where N is the number of villages, S $(1 \leq S \leq N)$ is the number of shops, Q is the number of queries to your program, and E $(1 \leq E \leq N)$ is the village you have to reach in order to leave the valley.

Each of the following $N - 1$ lines consists of three integers A, B, and W. This means that there is a road of length W $(1 \leq W \leq 10^9)$ connecting villages A and B $(1 \leq A \leq N, 1 \leq B \leq N)$.

Then S lines follow, each consisting of a single integer C, meaning that there is a shop in village C $(1 \leq C \leq N)$. Note that all of these lines are different, i.e. there is never more than one shop in a village.

Finally, there are Q lines, each containing two integers I and R, meaning that the I-th road from the input $(1 \leq I < N$, numbered in the order they are listed) is no longer usable and you want to know if your friends in village R $(1 \leq R \leq N)$ can leave the valley and if not, how far the closest village with a shop is.

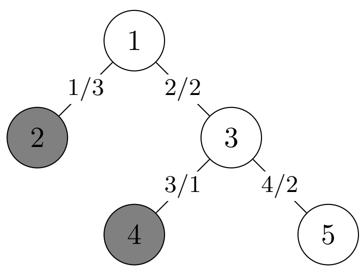
Output. Your output should consist of Q lines. The i-th line should contain the answer to the i-th query from the input. More precisely, the respective line should contain the string "`escaped`" (without quotes) if it is possible to leave the valley; if not, then it should contain the distance to the closest village with a shop, or the string "`oo`" if no shop is reachable anymore.

Example. Input

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

Output

```
escaped
3
oo
```

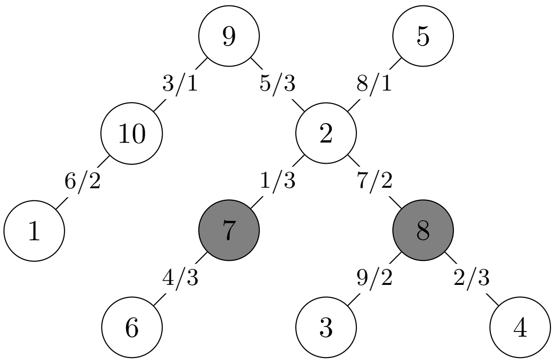


The figure on the right depicts the situation before a road becomes unusable. The villages with shops have been shaded in grey. The roads are labeled with "index / length". Exit from the valley is in village 1.

Example.

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

Output
```
8
escaped
escaped
escaped
0
```



Grading. The test groups satisfy the following conditions:

1. (9 points) $1 \le N \le 100$, $1 \le Q \le 10\,000$, and there is a road connecting villages A and B if and only if $|A - B| = 1$.
2. (27 points) $1 \le N \le 1\,000$, $1 \le Q \le 1\,000$.
3. (23 points) $1 \le N \le 100\,000$, $1 \le Q \le 100\,000$, and $S = N$.
4. (41 points) $1 \le N \le 100\,000$, $1 \le Q \le 100\,000$.