3D Labyrinth

You are in a 3D labyrinth of Conflict Empire and you want to escape it. The labyrinth could be represented as a three dimensional array where every cell is either empty or filled by rock. In every minute, you can travel in any direction parallel to the edges of the labyrinth, i.e. north, south, east, west, up and down, by one cell. You are not allowed to step onto a rock cell nor allowed to travel out of the labyrinth without reaching the exit point.

Given the starting point and the exit point, your task is to determine the minimum time needed to escape this labyrinth.

Input

The first line of the input contains three integers K, N and M (1 <= K,N,M <= 30), indicating the number of layers, rows and columns of the labyrinth, respectively. Then there follows K blocks of N lines each containing M characters. Each character describes one cell of the labyrinth, where `#` indicates a cell of rock and `.` indicates an empty cell. The starting position is indicated by `S` and the exit by `E`. `S` and `E` never *overlap*, they are always at distinct locations. There is always exactly one starting position and one exit position.

There will be a single blank line after each level.

Output

Print one line `Escaped in x minute(s).` where `x` is the minimum time needed to escape this labyrinth, or `Trapped!` if it is impossible to escape.

Example 1

```
Input:
3 4 5
S....
.###.
.##..
###.#

#####
#####
##.##
##...

#####
#####
#.###
####E


Output:
Escaped in 11 minute(s).
```

Example 2

```
Input:
1 3 3
S##
#E#
###


Output:
Trapped!
```

Example 3

```
3 3 3
###
#S#
###

###
#.#
###

###
#E#
###

Escaped in 2 minute(s).
```

Domino Effect

A domino effect or chain reaction is the cumulative effect produced when one event sets off a chain of similar events. Yu Ishigami is playing a new virtual novel game: Love is War. In this game, there are `n` distinct events. And if event `i` is triggered, it will trigger event `E[i]` as well. And `E[i]`` will trigger next events and so on. Initially no event is triggered. Yu can choose an event to trigger and he wants to maximize the number of overall triggered events. Please tell him which one to choose.

Input

The first line contains an integer `n`, indicating the number of events, `1 <= n < 50000`. In the each of n lines, the `i`-th line contains a single integer `E[i]`, which means event `i` will trigger event `E[i]`, `1 <= E[i] <= n`.

Output

Output the event that Yu should choose to maximize the number of triggered events. If there are more than one such event, output the one with the smallest value.

Example 1

```
Input:
3
2
3
1

Output:
1
```

Example 2

```
Input:
4
2
1
2
3

Output:
4
```

Example 3

```
Input:
5
2
1
4
5
3

Output:
3
```

Knight Tour

An `RxC` chess board has some squares filled with water. A knight can move `M` squares horizontally and `N` squares vertically, or `M` squares vertically and `N` squares horizontally in a single step. The knight can jump from `(a,b)` to `(c,d)` if and only if `|a-c|=N` and `|b-d|=M` or `|a-c|=M` and `|b-d|=N`. A knight **cannot** jump to squares filled with water. Of course the knight can not jump out the chess board.

The knight starts at square (1,1) (The intersection of 1st row and 1st column).

A square `(i,j)` is *valid* if it can be reached by a knight from the start square at `(1,1)`` with finite number of jumps. For each valid square` (i,j) `, we count the number of valid squares` (x,y) `such that the knight can jump from` (x,y) `to` (i,j)` in one step. If the count is even, we say (i,j) is an *even valid square*, otherwise we say (i,j) is an *odd valid square*.

Please count the number of *even valid squares* and *odd valid squares*.

Input

The first line contains four integer `R`, `C`, `M`, `N`, `1 < R,C <= 100`, `0 <= M,N <= 50`, and `M+N > 0`. The next line contains an integer `W`, `0 <= W < R*C`, which is the number of distinct squares filled with water. Each of the next `W` lines contain a pair of integer `x_i`, `y_i`, `1<=x_i<=R`, `1<=y_i<=C`, and `xi + yi > 2`, indicating that the square `(x_i,y_i)` is filled with water.

Output

Output two integer separated by space: the number of *even* and *odd valid squares*.

Example 1

```
Input:
3 3 2 1
0

Output:
8 0
```

(All the squares except for the center one are *valid*. Each of them can be accessed from two other squares, so they are all *even*.)

Example 2

```
Input:
4 4 1 2
2
4 4
2 2

Output:
4 10
```

Strange Race

Saya participated in a strange race during her winter break. In this race, each participant is given `N` cubes placed along a straight line from left to right. Each of those cubes is labeled with one distinct integer. The goal of this race is to rearrange the cubes as fast as possible such that the numbers on the cubes appear in increasing order from left to right after the rearrangement. To rearrange the cubes, the only allowed action is to swap the position of two consecutive cubes. Now, Saya is pondering what is the minimum number of moves required to achieve the goal. Your task is to help her out.

Input

The first line of the input contains one integers `N`, `1 <= M <= 1,000,000`, the number of cubes. The following `N` lines describe those cubes from left to right. Each of those `N` lines contains one integer `x`, `1 <= x <= 1,000,000,000`, indicating the number on that cube. It is guaranteed that there is no duplicate cube numbers.

Output

Print one line, containing a single integer indicating the minimum number of moves required to complete this race.

Example 1

```
Input:
3
3
1
2

Output:
2
```

Word play

Alan loves playing with words. He not only likes words that make sense but loves any sequence of alphanumeric characters. This time Alan is presented with two such sequences and he wants to find a subsequence which is common to both given sequences and has the maximum length possible.

Input

The input contains pairs of sequences. The first line of a pair is the first sequence and the second line contains the second sequence. The length of each sequence is at most 1000.

Output

For each pair of input lines, output a line containing one integer: the length of the maximal subsequence that both sequences have in common.

Example 1

```
Input:
bcacbcabbaccbab
bccabccbbabacbc
a1b2c3d4e
zz1yy2xx3ww4vv
abcdgh
aedfhr
abcdefghijklmnopqrstuvwxyz
a0b0c0d0e0f0g0h0i0j0k0l0m0n0o0p0q0r0s0t0u0v0w0x0y0z0
abcdefghijklmnzyxwvutsrqpo
opqrstuvwxyzabcdefghijklmn


Output:
11
4
3
26
14
```