TASK	OREHNJACA	ESEJ	VOYAGER	RAZLIKA	DLAKAVAC	AKVARIJ

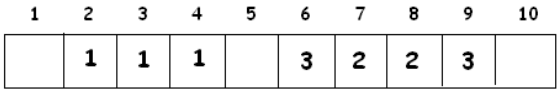
source code	`orehnjaca.pas`
	`orehnjaca.c`
	`orehnjaca.cpp`
		`esej.pas`
		`esej.c`
		`esej.cpp`
			`voyager.pas`
			`voyager.c`
			`voyager.cpp`
				`razlika.pas`
				`razlika.c`
				`razlika.cpp`
					`dlakavac.pas`
					`dlakavac.c`
					`dlakavac.cpp`
						`akvarij.pas`
						`akvarij.c`
						`akvarij.cpp`

input	standard input (*stdin*)					
output	standard output (*stdout*)					
time limit	1 second	1 second	1 second	0.5 seconds	2 seconds	1 second
memory limit	32 MB	32 MB	32 MB	64 MB	32 MB	256 MB
point value	**50**	**90**	**90**	**120**	**140**	**160**
	650					

Problems translated from Croatian by: **Ivan Pilat**

These days, the TV studio has started shooting another new season of Jamie Oliver's cooking show. This season, Jamie plans to introduce the delights of Croatian cuisine to the world. In the first episode, the master chef has baked a walnut roll **L meters** long, the longest ever baked in this part of the world. After hours of sweating and toiling in the kitchen, he has decided to reward each one of his **N faithful spectators** in the studio.

He has chopped the walnut roll into one meter long chops and marked them with numbers from 1 do **L**, from left to right. Each spectator has received a unique number ID (a positive integer from 1 to **N**), as well as a paper with two integers, **P** and **K**. Each spectator was then allowed to take all chops from the **P**-th to the **K**-th, inclusive. Spectators were allowed to take their share in order of their ID numbers (spectator 1 first, followed by spectator 2, etc.). This order resulted in some spectators receiving fewer chops than they initially thought they would get. The following image corresponds to the first example test case:



Write a program to determine which spectator **expected** to get the most walnut roll chops, and which spectator **actually got** the most.

INPUT

The first line of input contains the positive integer **L** ($1 \leq \mathbf{L} \leq 1000$), the length of the walnut roll.
The second line of input contains the positive integer **N** ($1 \leq \mathbf{N} \leq 1000$), the number of spectators.
Each of the following **N** lines contains two positive integers $\mathbf{P_i}$ and $\mathbf{K_i}$ ($1 \leq \mathbf{P_i} \leq \mathbf{K_i} \leq \mathbf{L}$, $\mathbf{i} = 1..\mathbf{N}$), the values **P** and **K** as described in the problem statement for spectator number **i**.

OUTPUT

The first line of output must contain the ID number of the spectator who was expecting to receive the most walnut roll chops.

The second line of output must contain the ID number of the spectator who actually received the most walnut roll chops in the end.

In both cases, if there is more than one spectator satisfying the condition, output the one with the smallest ID.

SCORING

If the first number is correct, the solution is awarded 60% of points for that test case, and if the second number is correct, the solution is awarded 40% of points for that test case.

SAMPLE TESTS

input	**input**	**input**
10 3 2 4 7 8 6 9	10 3 1 3 5 7 8 9	10 5 1 1 1 2 1 3 1 4 7 8
output	**output**	**output**
3 1	1 1	4 5

Mirko's latest homework assignment is writing an essay. However, he finds writing essays so boring that, after working for two hours, he realized that all he has written are **N** long words consisting entirely of letters A and B. Having accepted that he will never finish the essay in time, poor Mirko has decided to at least have some fun with it by counting **nice** words.

Mirko is connecting pairs of **identical** letters (A with A, B with B) by drawing arches **above** the word. A given word is nice if each letter can be connected to exactly one other letter in such a way that no two arches intersect. Help Mirko count how many words are nice.

INPUT

The first line of input contains the positive integer **N** ($1 \leq \mathbf{N} \leq 100$), the number of words written down by Mirko.

Each of the following **N** lines contains a single word consisting of letters A and B, with length between 2 and 100 000, inclusive. The sum of lengths of all words doesn't exceed 1 000 000.

OUTPUT

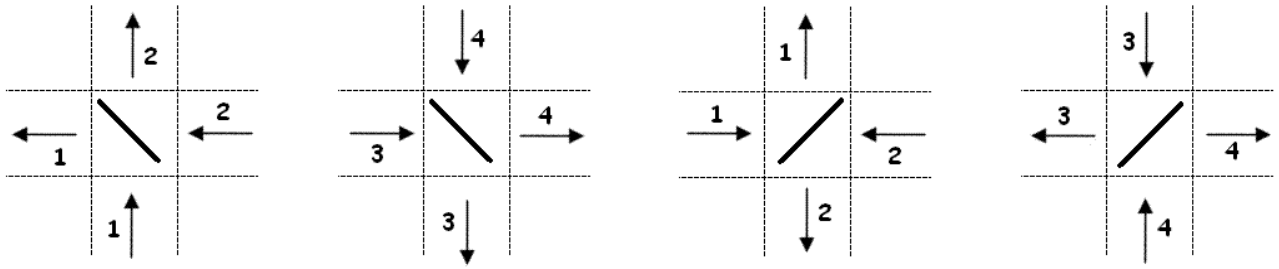
The first and only line of output must contain the number of nice words.

SAMPLE TESTS

input	**input**	**input**
3 ABAB AABB ABBA	3 AAA AA AB	1 ABBABB
output	**output**	**output**
2	1	1

The Voyager 1 space probe (not to be confused with the Intrepid-class starship) was launched a long time ago, in 1977, and is currently on the verge of leaving our Solar System. As it travels further through space, it has been programmed to leave a radio signal message in any star system it stumbles upon, to mark the probe's path for as long as possible.

Let us assume that a star system can be represented by a rectangular grid with **N rows** and **M columns**, dividing the space into **N by M equal cells**. Each cell can contain a single **planet**, **black hole**, or be **empty**. The probe broadcasts the signal from a pre-determined **empty cell**, in one of the four axis-aligned directions ("U"-up, "R"-right, "D"-down, "L"-left).
Upon being broadcast, the signal propagates in a straight line along the same row/column until it reaches a planet, where it is deflected by 90 degrees in another direction. There are two kinds of planets, which we will denote by „/" and „\". The deflection rules are shown in the image below:



The signal permanently leaves the system upon either entring a cell containing a black hole, or propagating outside the edges of the rectangular grid. It is also known that the signal needs **one second** to propagate from the current cell to a neighbouring one.

Write a program to determine the direction in which the probe needs to broadcast the signal so that it remains within the system for **as long as possible**, outputting the optimal direction as well as the resulting longest time. If it is possible for the signal to remain in the system indefinitely, output the message „Voyager" instead of the required time.

INPUT

The first line of input contains two positive integers, **N** ($1 \leq \mathbf{N} \leq 500$) and **M** ($1 \leq \mathbf{M} \leq 500$).

Each of the following **N** lines contains **M** characters from the set {"/", "\", "C", "."}, where "/" and "\" represent the two kinds of planets, "C" represents a black hole, and "." represents an empty cell.

The last line of input contains two positive integers, **PR** ($1 \leq \mathbf{PR} \leq \mathbf{N}$) and **PC** ($1 \leq \mathbf{PC} \leq \mathbf{M}$), the row and column number, respectively, of the cell where the probe is situated.

OUTPUT

The first line of output must contain the required optimal broadcast direction ("U", "R", "D", or "L"). If the solution is not unique, select the first optimal one in the following priority order: first "U", then "R", then "D", and finally "L".

The second line of output must contain the required longest time (or message).

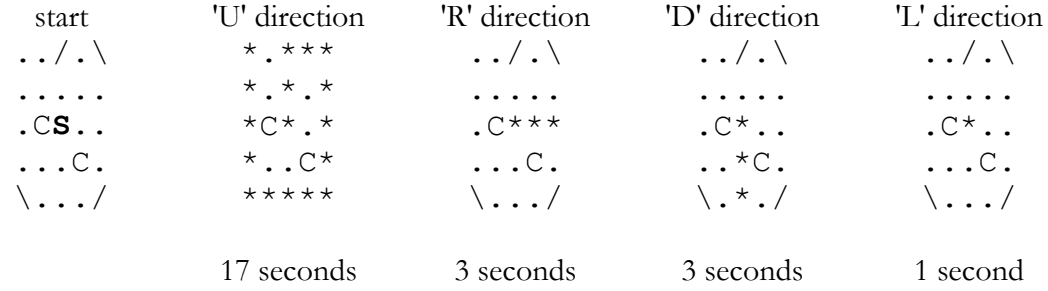
SCORING

In test data worth at least 50% of total points, the signal will not be able to remain in the system indefinitely.

SAMPLE TESTS

input	**input**	**input**
5 5 ../.\ ..... .C... ...C. \.../ 3 3	5 5 ....\ \..\. ./\.. \../C .\../ 1 1	5 7 /.....\ ../..\. \...../ /.....\ \.\.../ 3 3
output	**output**	**output**
U 17	D 12	R Voyager

Clarification of the first example ("*" represents the path of the singal)**:**



Mirko's newest math homework assignment is a very difficult one! Given a sequence, **V**, of **N** integers, remove exactly **K** of them from the sequence. Let **M** be the largest difference of any two remaining numbers in the sequence, and **m** the smallest such difference. Select the **K** integers to be removed from **V** in such a way that the sum **M** + **m** is the smallest possible. Mirko isn't very good at math, so he has asked you to help him!

INPUT

The first line of input contains two positive integers, **N** ($3 \leq \mathbf{N} \leq 1\,000\,000$) and **K** ($1 \leq \mathbf{K} \leq \mathbf{N} - 2$).

The second line of input contains **N** space-separated positive integers – the sequence **V** ($-5\,000\,000 \leq \mathbf{V}_i \leq 5\,000\,000$).

OUTPUT

The first and only line of output must contain the smallest possible sum **M** + **m**.

SAMPLE TESTS

input	**input**	**input**
5 2 -3 -2 3 8 6	6 2 -5 8 10 1 13 -1	6 3 10 2 8 17 2 17
output	**output**	**output**
7	13	6

In the faraway city of Xanadu, a flu epidemic has broken out, caused by a strain known as hairy flu. There are **M** people living in the city, each resident having a unique personal ID number from the range of 0 to **M** – 1, inclusive. Infection with this strain lasts exactly one day, and a person can catch it multiple times per season (since it mutates too quickly for lasting immunity).

On the first day of the epidemic, the flu was brought from another faraway country by a group of residents nicknamed "**init-patients**", whose ID numbers are known. The flu's spread is based on them.

Each following day, a resident with ID number **p** will catch the flu iff there exists a resident with ID **a** who was infected the previous day, as well as an init-patient with ID **b**, such that:

$$(\mathbf{a} * \mathbf{b}) \bmod \mathbf{M} = \mathbf{p}.$$

The numbers **a** and **b** need not be distinct. For example, consider a case where there are 101 people in the town, and the init-patients are 5 and 50. On the first day, the init-patients are infected by definition. On the second day, the residents infected are 25, 48 (250 mod 101), and 76 (2500 mod 101). On the third day, one of the infected patients is 77, since (48 * 50) mod 101 = 77.

Who will catch the flu on the **K**-th day?

INPUT

The first line of input contains three positive integers, **K**, **M**, and **N** ($1 \leq \mathbf{K} \leq 10^{18}$, $3 \leq \mathbf{M} \leq 1500$, $\mathbf{N} < \mathbf{M}$).

The second line of input contains **N** space-separated nonnegative integers, the personal ID numbers of residents who were infected on the first day (the init-patients). These numbers are unique, increasing, and do not exceed **M** - 1.

OUTPUT

The first and only line of output must contain the personal ID numbers of residents infected with flu on the **K**-th day, given space-separated and in increasing order.

SAMPLE TESTS

input	**input**	**input**
1 100 3 1 2 3	2 100 3 1 2 3	10 101 2 5 50
output	**output**	**output**
1 2 3	1 2 3 4 6 9	36 44 57 65

Mirko has recently installed a new screensaver. If he is away from the keyboard for five minutes, the screen shows a picture of an aquarium with animated fish. The screensaver has settings for customizing the shape of the (virtual, sandy) aquarium bottom, as well as the water level.

The aquarium can be represented in a 2D Cartesian coordinate system as a shape **N** - 1 columns wide, where **N** is a positive integer. The left wall of the aquarium has the x-coordinate of 0, and the right wall has the x-coordinate of **N** - 1. Each integer-valued x-coordinate of the aquarium bottom (let us denote it by **i**) from 0 to **N** - 1 has a separately adjustable height of $\mathbf{H}_i$. Between any two adjacent integer-valued x-coordinates **i** and **i** + 1, the bottom can be described by a line segment between points (**i**, $\mathbf{H}_i$) and (**i** + 1, $\mathbf{H}_{i+1}$).

If the water level is set to **h**, the water fills the area between the line **y** = **h** and the aquarium bottom. If a part of the aquarium bottom is above the water level **h**, it forms an island and is not submerged.

For different shapes of the aquarium bottom, Mirko would like to know the total area of his screen covered by water. Help Mirko find answers to his questions (other than 42).

INPUT

The first line of input contains two positive integers, **N** ($3 \leq \mathbf{N} \leq 100\,000$, the length of the bottom) and **M** ($1 \leq \mathbf{M} \leq 100\,000$, the number of queries).

The second line of input contains **N** space-separated nonnegative integers $\mathbf{H}_i$ ($0 \leq \mathbf{H}_i \leq 1000$), the starting bottom heights.

Each of the following **M** lines contains a single query with one of the following two types:

Q **h** – if the water level is set to **h** ($0 \leq \mathbf{h} \leq 1000$), assuming the current bottom shape, what is the total screen area covered by water?

U **i h** – Mirko has decided to change the bottom height at x-coordinate **i** ($0 \leq \mathbf{i} \leq \mathbf{N} - 1$) to **h** ($0 \leq \mathbf{h} \leq 1000$); in other words, set $\mathbf{H}_i = \mathbf{h}$.

OUTPUT

For each query with type Q, output a single line containing the required area, rounded to exactly three decimals. The area given is allowed to differ by at most 0.001 from the official solution.

SAMPLE TESTS

input	input	input
3 2 20 20 20 Q 20 Q 30	3 5 0 2 0 Q 2 U 1 1 Q 1 U 1 10 Q 5	7 7 0 2 1 3 2 1 0 Q 1 Q 2 Q 3 U 3 0 Q 1 Q 2 Q 3
output	**output**	**output**
0.000 20.000	2.000 1.000 2.500	0.750 3.750 9.000 1.500 6.000 12.000

Clarification of the third example: The left image below shows the situation before, and the right one after the U-type query, for water level **h** = 2 (query Q 2). In the first image, the submerged area equals 3.75, and in the second image it is 6.