

TASK	MAJSTOR	NIZOVI	TABLICA	CVJETICI
<b>input</b>	standard input			
<b>output</b>	standard output			
<b>time limit</b> (per test case)	1 second			
<b>memory limit</b> (per test case)	32 MB			
<b>points</b>	<b>35</b>	<b>45</b>	<b>55</b>	<b>65</b>
	<b>200</b>			

Rock-paper-scissors is a popular two-player game. In the game, each of the players uses their hand to show one of three symbols: rock, paper or scissors. If both players show the same symbol, the game is a tie. Otherwise, scissors beat paper, paper beats rock and rock beats scissors.

Sven has been studying the psychological intricacies of the game for years and has become a real master at the game, his friends not standing a chance against him in one-on-one games.

With the world championships around the corner, Sven is practicing his skills playing simultaneous games with  $N$  of his friends. One such game consists of  $R$  rounds. In each round, Sven and each of his friends show one of the three symbols.

When calculating the score, in each round, Sven's symbol is independently compared to each of his friends' symbols. Sven scores two points for every win and one point for every tie. Sven does not get points for losing.

Write a program that calculates Sven's total score, and also his largest possible score had he known in advance all the symbols his friends would show.

### INPUT

The first line contains the integer  $R$  ( $1 \leq R \leq 50$ ), the number of rounds played.

The second line contains a string of  $R$  letters 'S', 'P' or 'R'. The string represents symbols that Sven showed in each round. 'S' is for scissors, 'P' for paper, 'R' for rock.

The third line contains the integer  $N$  ( $1 \leq N \leq 50$ ), the number of friends.

Each of the following  $N$  lines contains a string of  $R$  letters 'S', 'P' or 'R'. These are the symbols shown by each of the  $N$  friends in each of the  $R$  rounds.

### OUTPUT

Output Sven's actual score on the first line.

Output his largest possible score on the second line, assuming his friends didn't change their symbols.

### EXAMPLES

<b>input</b> 5 SSPPR 1 SSPPR  <b>output</b> 5 10	<b>input</b> 5 SSPPR 2 PPRRS RRSSP  <b>output</b> 10 15	<b>input</b> 4 SPRS 4 RPRP SRRR SSPR PSPS  <b>output</b> 12 21
--	--	---

Marko's math notebook has two sequences of integers written in it, both of length  $N$ . The **fuzziness** of these sequences is calculated by first **reversing** the order of the elements in the second sequence and then adding the **products** of elements in the same positions of the two sequences.

3	-4	-3	-2	2	0
-3	0	5	-1	3	2

For example, the fuzziness of the above two sequence of length 6 is  $3 \cdot 2 + (-4) \cdot 3 + (-3) \cdot (-1) + (-2) \cdot 5 + 2 \cdot 0 + 0 \cdot (-3) = -13$ .

Mirko likes his pairs of sequences to be as fuzzy as possible. He decided to remove  $B$  numbers (possibly zero) from the beginning of both sequences and also  $E$  numbers (possibly zero) from the end of both sequences, so that the fuzziness is as large as possible.

Write a program that finds the values of  $B$  and  $E$  for which the fuzziness is largest.

### INPUT

The first line of input contains the integer  $N$  ( $1 \leq N \leq 2000$ ), the length of Marko's sequences.

The following two lines contain  $N$  integers each, the two sequences. All numbers in the two sequences will be between  $-1000$  and  $1000$ .

### OUTPUT

Output two integers  $B$  and  $E$  on the first line, such that  $0 \leq B, E < N$  and  $B+E < N$ .

Output the resulting fuzziness on the second line.

If multiple choices of  $B$  and  $E$  achieve the largest fuzziness, output any of them.

### SCORING

In test data worth a total of 30 points,  $N$  will be at most 100.

### EXAMPLES

<b>input</b> 6 3 -4 -3 -2 2 0 -3 0 5 -1 3 2  <b>output</b> 0 3 24	<b>input</b> 5 1 1 1 1 1 2 2 2 2 2  <b>output</b> 0 0 10	<b>input</b> 5 5 -5 -5 -5 5 -5 -5 5 -5 -5  <b>output</b> 2 0 75
--	---	--

Ivo has an  $N \times N$  table. The table has the integers 1 through  $N^2$  inscribed in row-major order. The following operations can be done on the table:

1. Rotate a row – all cells in a single row are rotated right, so that the number in the last column moves to the first.
2. Rotate a column – all cells in a single column are rotated down, so that the number in the last row moves to the first.

Ivo occasionally feels the urge to move a number  $X$  to cell  $(R, C)$  and proceeds as follows:

- While  $X$  is not in column  $C$ , rotate the row it is in.
- While  $X$  is not in row  $R$ , rotate the column it is in.

Here is an example of how to move number 6 to cell  $(3, 4)$ , start from the initial configuration:

1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	16
5	6	7	8	8	5	6	7	7	8	5	6	7	8	5	4
9	10	11	12	9	10	11	12	9	10	11	12	9	10	11	6
13	14	15	16	13	14	15	16	13	14	15	16	13	14	15	12

Ivo wants to move  $K$  numbers one after another. Write a program that calculates the number of rotations needed.

### INPUT

The first line contains two integers  $N$  ( $2 \leq N \leq 10000$ ) and  $K$  ( $1 \leq K \leq 1000$ ), the table dimension and the number of moves.

Each of the following  $K$  lines contains three integers  $X$  ( $1 \leq X \leq N^2$ ),  $R$  and  $C$  ( $1 \leq R, C \leq N$ ), the description of one move Ivo wants to make. Ivo does the moves in the order in which they are given.

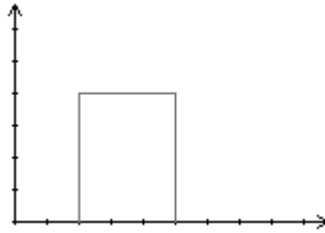
### OUTPUT

Output  $K$  lines; for each move, output the number of rotations needed.

### EXAMPLES

<b>input</b> 4 1 6 3 4  <b>output</b> 3	<b>input</b> 4 2 6 3 4 6 2 2  <b>output</b> 3 5	<b>input</b> 5 3 1 2 2 2 2 2 12 5 5  <b>output</b> 2 5 3
--	--	---

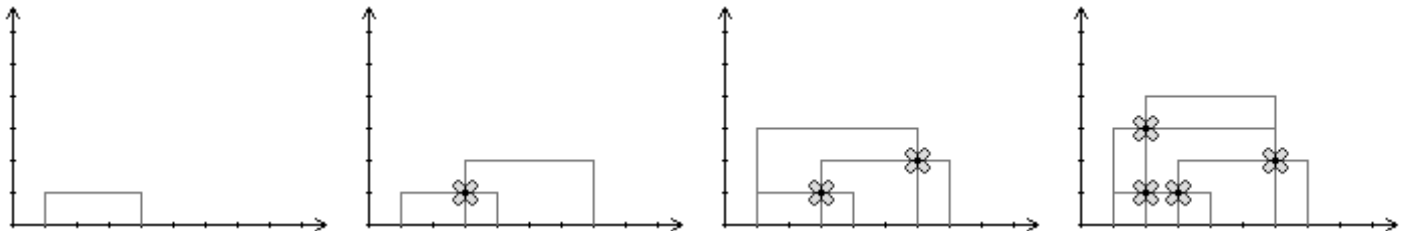
On a faraway planet, strange plants with two stems can be found. Every plant on the planet can be described by three numbers: the x-coordinates of the stems L and R, and the height H at which the stems are connect. The image depicts a plant with L=2, R=5 and H=4.



Every day a new plant grows on the planet. The plant that grows on day 1 is of height 1, and every subsequent plant is one higher than the previous one.

When a stem of a new plant **intersects** the **horizontal** segment of another plant, a small flower grows (if one wasn't there already). If segments merely touch in a point, a flower will not grow there.

The following images are a visualization of the first example on the next page.



Write a program that, given the coordinates of all plants, calculates the number of new flower every day.

### INPUT

The first line contains an integer  $N$  ( $1 \leq N \leq 100\,000$ ), the number of days.

Each of the following  $N$  lines contains two integers  $L$  and  $R$  ( $1 \leq L < R \leq 100\,000$ ), the coordinates of the stems of a plant.

### OUTPUT

Output  $N$  lines, the number of new flowers after each plant grows.

EXAMPLES

<p>input</p> <p>4</p> <p>1 4</p> <p>3 7</p> <p>1 6</p> <p>2 6</p> <p>output</p> <p>0</p> <p>1</p> <p>1</p> <p>2</p>	<p>input</p> <p>5</p> <p>1 3</p> <p>3 5</p> <p>3 9</p> <p>2 4</p> <p>3 8</p> <p>output</p> <p>0</p> <p>0</p> <p>0</p> <p>3</p> <p>2</p>
---	---