CROATIAN OPEN COMPETITION IN INFORMATICS

5th ROUND

COCI 2009/2010 **Task SOK**

5th round, 6. March 2010. 1 second / 32 MB / 30 points

Mirko and Slavko bought a few litters of orange, apple and pineapple juice. They are now whipping up a non alcoholic cocktail following a recipe they found on the Internet. Sadly, they figured out too late that not only you should use recipes when making cocktails, you should also use them when planning how much juice to buy.

Write a program that will determine how much of each juice they will have leftover, after they make as much cocktail as possible, respecting the recipe.

INPUT

The first line contains three integers, **A**, **B**, **C**, ($1 \leq$ **A**, **B**, **C** ≤ 500), amount of orange, apple and pineapple juice they bought, in liters.

The second line contains three integers, **I**, **J**, **K**, ($1 \leq$ **I**, **J**, **K** ≤ 50), the ratio of orange, apple and pineapple juice in the cocktail.

OUTPUT

The first and only line of output should contain three decimal numbers, leftover amounts of each juice, in liters.

Note: solutions with relative error 10^{-4} or smaller will be accepted.

SAMPLE TEST CASES

Input:	**Input:**	**Input:**
10 10 10 3 3 3	9 9 9 3 2 1	10 15 18 3 4 1
Output:	**Output:**	**Output:**
0 0 0	0 3 6	0 1.666667 14.666667

COCI 2009/2010 **Task CUDOVISTE**

5th round, 6. March 2010. 1 second / 32 MB / 50 points

Mirko got his drivers license! To celebrate that joyous occasion, his parents bought him his first car: a monster truck! Mirko found out that even though having a car that can squash all other cars is nice in traffic jams, parking a car that is the size of 4 normal cars can be a bit tricky.

His friend, Slavko, works part time in the city parking company. He periodically sends Mirko a map of the city with occupied parking spaces marked. The map can be represented as a table with **R** rows, **C** columns each. Each cell can contain a building (symbol '#'), a parked car (symbol 'X') or a free parking space (symbol '.'). A monster truck is quite huge, **2 by 2 cells** to be exact.

Help Mirko calculate the number of possible parking space grouped by the number of cars he needs to squash to park in them. We are only interested in the number of cars Mirko will squash on the parking space, **not the number of cars he will squash on the way over**. However, Mirko can't park on a building. Not even a monster truck is large enough to squash buildings!

INPUT

The first line of input contains two integers, **R** and **C** ($2 \leq$ **R**, **C** ≤ 50), the number of rows and columns of the map.

The second R lines contain C characters each. Only characters '#', 'X' and '.' appear in the input. Note that 'X' will always be capital.

OUTPUT

The output consists of five lines, the total number of parking spaces Mirko can park on if he squashes 0 cars (first line), 1 car (second line), 2 cars (third line), 3 cars (fourth line), 4 cars (fifth line).

SAMPLE TEST CASES

Input:	**Input:**	**Input:**
4 4 #..# ..X. ..X. #XX#	4 4	4 5 ..XX. .#XX. ..#.. .....
Output:	**Output:**	**Output:**
1 1 2 1 0	9 0 0 0 0	2 1 1 0 1

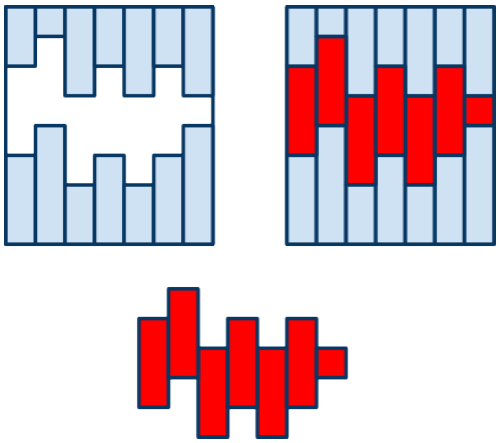
COCI 2009/2010 **Task KLETVA**

5th round, 6. March 2010. 1 second / 32 MB / 70 points

As punishment for destroying half of his city with his monster truck, Mirko now has to pay off his dept to society. He works as an assistant for a famous archaeologist. One of his duties include crafting keys for ancient document boxes.

In ancient times document boxes were locked using elaborate mechanisms with interesting locks. Each lock is **L** centimeters long and **W** centimeters wide and consists of three parts, the upper edge, the lower edge and the empty area between them. Both edges can be represented as a sequence of **L nonnegative integers**: r_1 r_2 r_3 ... r_L. Each number in sequence represents the width of edge at that point.

The key for each lock is a small clay tab, fitting perfectly in the area between edges. This image shows a 7 cm long, 8 cm wide lock along with the corresponding key.



The sequence representing the upper edge is [2, 1, 3, 2, 3, 2, 3], and the sequence representing the lower edge is [3, 4, 2, 3, 2, 3, 4]. Mirko noticed that some keys open more than one lock. Making keys is tedious work so Mirko asked you to find out what is the **minimal number** of different keys he needs to make and still be able to open all of the locks.

INPUT

First line of input contains three integers, **W** ($1 \leq \mathbf{W} \leq 10^8$), width of all locks, **L** ($1 \leq \mathbf{L} \leq 1000$) length of all locks, and **N** ($1 \leq \mathbf{N} \leq 100$), number of different locks.

Next 2**N** lines describe all locks. Each line contains exactly **L** numbers smaller than **W**. Each pair of lines describes one lock. The first line in one pair describes the upper edge, and the second line the lower edge. **There shall always be at least 1 cm of empty space between both edges on all locks.**

OUTPUT

The first and only line of input should contain a single integer, the minimal number of different keys Mirko needs to craft.

SAMPLE TEST CASES

Input:	**Input:**	**Input:**
8 7 2 2 1 3 2 3 2 3 3 4 2 3 2 3 4 3 2 4 3 4 3 4 2 3 1 2 1 2 3	8 4 4 3 3 3 3 3 3 3 3 2 2 2 2 4 4 4 4 1 2 3 4 4 3 2 1 1 1 1 1 5 5 5 5	100000000 2 2 88888888 88888888 4 4 4 4 88888888 88888888
Output:	**Output:**	**Output:**
1	2	1

COCI 2009/2010 **Task ZUMA**

5th round, 6. March 2010. 1 second / 32 MB / 100 points

One day Mirko, while he was walking through the high grass, stumbled upon a sequence of N colored marbles. Soon he noticed that if he touches **K or more consecutive** marbles of the **same color**, they start to twinkle and then he could wish them to magically vanish, although he doesn't have to do that immediately (see 3. sample). Fortunately, Mirko brought an inexhaustible supply of marbles from home, so he can insert a marble of any color anywhere in the array (at the beginning, between any two existing marbles, or at the end). Help Mirko find the smallest number of marbles he must insert into the sequence before he could make all of the marbles vanish.

INPUT

The first line of input contains two integers **N** ($1 \leq \mathbf{N} \leq 100$) and **K** ($2 \leq \mathbf{K} \leq 5$) - the number of marbles in the initial sequence and the minimal number of consecutive marbles of the same color he could wish to vanish.
The next line contains exactly **N** integers between 1 and 100 (inclusive), separated by one space. Those numbers represent colors of marbles in the sequence Mirko found.

OUTPUT

The output should contain only one line with a single integer number - the minimal number of marbles Mirko has to insert to achive the desired effect.

SAMPLE TEST CASES

Input:	**Input:**	**Input:**
2 5 1 1	5 3 2 2 3 2 2	10 4 3 3 3 3 2 3 1 1 1 3

Output:	**Output:**	**Output:**
3	2	4

COCI 2009/2010 **Task PROGRAM**

5th round, 6. March 2010. 5 seconds / 32 MB / 120 points

Mirko is trying to debug a piece of his code. First he creates an array of N integers and fills it with zeros. Then he repeatedly calls the following procedure (he is such a good coder he coded it in both C++ and Pascal):

```
void something( int jump ) {
   int i = 0;
   while( i < N ) {
      seq[i] = seq[i] + 1;
      i = i + jump;
   }
}
```

```
procedure something( jump: longint );
var i : longint;
begin
  i := 0;
  while i < N do
  begin
    seq[i] := seq[i] + 1;
    i := i + jump;
  end;
end;
```

As you can see, this procedure increases by one all elements in the array whose indices are divisible by `jump`.

Mirko calls the procedure exactly **K** times, using the sequence $\mathbf{X_1}\ \mathbf{X_2}\ \mathbf{X_3}\ldots\ \mathbf{X_k}$ as arguments.

After this, Mirko has a list of **Q** special parts of the array he needs to check to verify that his code is working as it should be. Each of this parts is defined by two numbers, **L** and **R** (**L** ≤ **R**) the left and right bound of the special part. To check the code, Mirko must compute the sum of all elements of `seq` between and including **L** and **R**. In other words `seq`[**L**] + `seq`[**L**+1] + `seq`[**L**+2] + ... + `seq`[**R**]. Since he needs to know the answer in advance in order to check it, he asked you to help him.

INPUT

The first line of input contains two integers, **N** ($1 \le \mathbf{N} \le 10^6$), size of the array, and **K** ($1 \le \mathbf{K} \le 10^6$), number of calls to `something` Mirko makes.

The second line contains **K** integers: $\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3$ ... $\mathbf{X}_k$, arguments passed to the procedure. ($1 \leq \mathbf{X}_i < \mathbf{N}$).
Next line contains one integer **Q** ($1 \leq \mathbf{Q} \leq 10^6$), number of special parts of the array Mirko needs to check.
Next **Q** lines contain two integers each $\mathbf{L}_i$ i $\mathbf{R}_i$($0 \leq \mathbf{L}_i \leq \mathbf{R}_i < \mathbf{N}$), bounds of each special part.

OUTPUT

The output should contain exactly **Q** lines. The i^{th} line should contain the sum of elements seq[$\mathbf{L}_i$] + seq[$\mathbf{L}_i$+1] + seq[$\mathbf{L}_i$+2] + ... + seq[$\mathbf{R}_i$].

SAMPLE TEST CASES

Input:	**Input:**	**Input:**
10 4 1 1 2 1 3 0 9 2 6 7 7	11 3 3 7 10 3 0 10 2 6 7 7	1000000 6 12 3 21 436 2 19 2 12 16124 692 29021
Output:	**Output:**	**Output:**
35 18 3	8 2 1	16422 28874

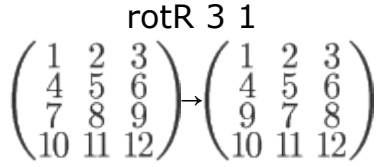
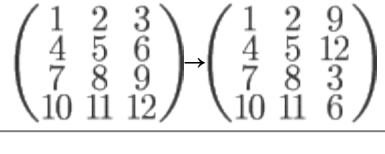
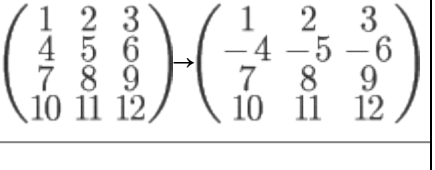
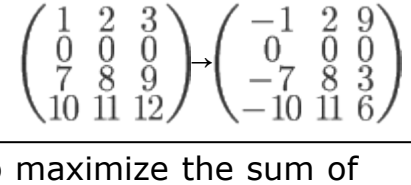
Sample 1. description: The procedure is called with arguments 1, 1, 2, 1. After that the array contains values {4, 3, 4, 3, 4, 3, 4, 3, 4, 3}. Sum of indices 2 to 6 (inclusive) is 4+3+4+3+4 = 18.

Sample 2. description: The array is {3, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1}.

COCI 2009/2010 **Task CHUCK**

5th round, 6. March 2010. 1 second / 32 MB / 130 points

You are given an matrix of **R** rows and **C** columns. All elements of the matrix are by their absolute value smaller than or equal to 10^4.
You may perform the following operations:

Operation	Notation	Example
Rotate ***i***-th row of the matrix ***k*** elements right	rotR i k	rotR 3 1 
Rotate ***j***-th column of the matrix ***k*** elements down	rotS j k	rotS 3 2 
Multiply all elements in the ***i***-th row by -1, **if and only if none of them were multiplied before.**	negR i	negR 2 
Multiply all elements in ***j***-th column by -1, **if and only if none of them were multiplied before.**	negS j	negS 1 

Using limited number of these operations, you need to maximize the sum of all the elements of the matrix.

INPUT

The first line of input contains two integers **R** and **C** ($1 \leq$ **R**, **C** ≤ 100), number of rows and columns.
The next **R** lines contain **C** integers each. All integers are by their absolute value smaller than 10^4.

OUTPUT

The first line of output should contain two integers, the maximal sum obtainable and the number of operations used. We shall call this number **T**. The next **T** lines should contain any sequence of operations leading to the sum. Each operation should follow the notation defined in the table below. For details look at sample test cases.

SCORING

- If the obtained sum is not maximal, one of the elements was multiplied more than once or the sequence of operations printed does not lead to the sum, 0 points are awarded.
- Otherwise, the number of points depends on the number of operations used
    - For **T** $\leq 5 \cdot R \cdot C$, you are awarded 100% of points allocated to that test case
    - For $5 \cdot R \cdot C$ < **T** $\leq$ 100 000, you are awarded 50% of points allocated to that test case
    - For **T** > 100 000, you are awarded 0 points for that test case

SAMPLE TEST CASES

Input:	**Input:**
3 4 1 -2 5 200 -8 0 -4 -10 11 4 0 100	3 3 8 -2 7 1 0 -3 -4 -8 3
Output:	**Output:**

345 2 rotS 2 1 negR 2	34 4 rotR 1 1 rotS 3 1 negR 2 negR 3