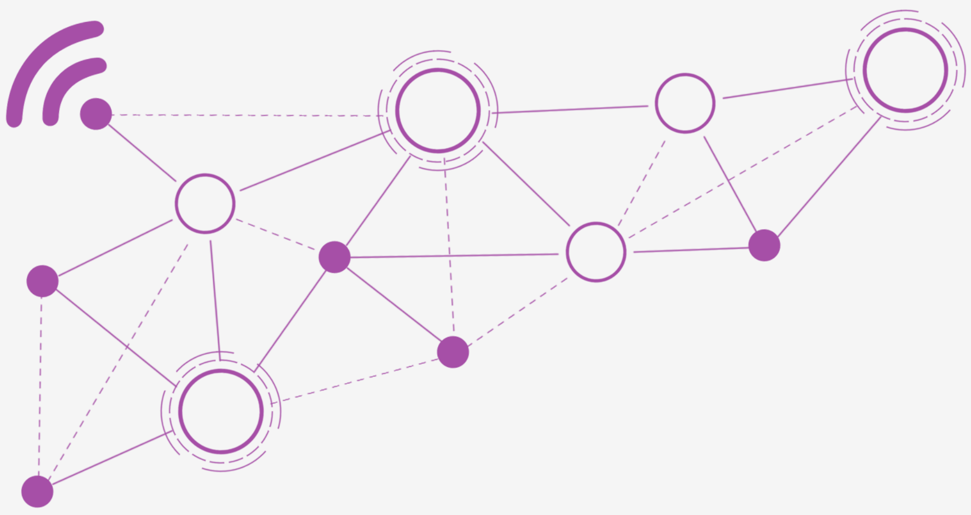


2023

National Olympiad in Informatics

Eliminations

Contents

Notes

- Many problems have large input file sizes, so use fast I/O. For example:
  - In Java, use `BufferedReader` and `PrintWriter`.
  - In C/C++, use `scanf` and `printf`.
- Good luck and enjoy the problems!

Important! Read the following:

- Your solution will be checked by running it against one or more (usually several) hidden test cases. You will not have access to these cases, but a correct solution is expected to handle them correctly.
- The output checker is **strict**. Follow these guidelines strictly:
  - It is **space sensitive**. Do not output extra leading or trailing spaces. Do not output extra blank lines unless explicitly stated.
  - It is **case sensitive**. So, for example, if the problem asks for the output in lowercase, follow it.
  - Do not print any tabs. (No tabs will be required in the output.)
  - Do not output anything else aside from what's asked for in the Output section. So, do not print things like "Please enter t".

  Not following the output format strictly and exactly will likely result in the verdict "*Output isn't correct*".
- Do not read from, or write to, a file. You must read from the standard input and write to the standard output.
- For Java, do not add a package line at the top of your code. Otherwise, your submission will receive the verdict "*Execution failed because the return code was nonzero*" or "*Compilation failed*".
- Only include one file when submitting: the source code (.cpp, .java, .py, etc.) and nothing else.
- Only use letters, digits and underscores in your filename. Do not use spaces or other special symbols.
- Many problems have large input file sizes, so use fast I/O. For example:
  - In Java, use `BufferedReader` and `PrintWriter`.
  - In C/C++, use `scanf` and `printf`.

  We recommend learning and using these functions during the Practice Session.
- Good luck and enjoy the contest!

Problem A
Sharing Candies
Time Limit: 1 second

At the store, there are seven packs of NOI.PH-brand candies, with each pack containing a possibly different amount of candies. Jack and Jill were instructed by their mother to buy **exactly** two packs of NOI.PH-brand candies, although they haven't yet decided on which to buy.

Jack and Jill want to show their mother that they are *nice* children, and so they have a plan. They will first combine all the candies between the two packs that they bought into one large pile. They will give one candy to their mother, and then split the remaining candies equally between the two of them.

Given the number of candies in each pack, determine if it is possible for Jack and Jill to carry out their plan if they buy the right packs of candies.

Input Format

Input consists of seven positive integers, the number of candies in each pack, each in its own line.

Output Format

If Jack and Jill can carry out their plan, output `Candies for everyone!` in its own line. Otherwise, output `No candy for Mom :(` instead.

Constraints and Subtasks

For all subtasks
Each pack has between 1 and 100 candies.

Subtask	Points	Constraints
1	**50**	Each pack has between 1 and 2 candies.
2	**30**	The first pack (given on the first line) contains 1 candy.
3	**20**	No further constraints

Input 1	Output 1
30 41 9 30 2 58 88	Candies for everyone!

Input 2	Output 2
30 4 20 30 2 58 88	No candy for Mom :(

Problem B
Hans gets Disqualified

Time Limit: 1 second

Hans got disqualified for breaking the NOI.PH rules (which you can read at `noi.ph/rules`*). It would be best for you to read the rules too so that you don't end up disqualified like Hans, even if you don't intend to cheat!*

This entire problem happens sometime in the future.

It turns out that Hans has actually been caught cheating in the past, as summarized in a report provided by his school. The NOI.PH Scientific Committee would like your help in determining *how many times* he's been caught cheating in the past. And please format your answer nicely!

Input Format

Input consists of a single line which contains a list of timestamps in the standard ISO format `yyyy-mm-dd hh:mm:ss`, with a comma and space `, ` between two timestamps. These timestamps each correspond to a cheating incident, and are all guaranteed to be unique.

Output Format

Output a single line containing the sentence,

```
There <'has'/'have'> been <count> <'incident'/'incidents'> of
                    cheating in the past.
```

(There is a single space between `of` and `cheating`; the line break here is just so that the line fits on the page.).

Here, replace `<count>` with the number of recorded cheating incidents, and choose the proper conjugation (singular or plural) of `<'has'/'have'>` and `<'incident'/'incidents'>`. See the Sample I/O for examples

Constraints and Subtasks

For all subtasks
The input contains at least 1 and no more than 500 characters (including spaces). Each timestamp in the input corresponds to a valid date and time.

Subtask	Points	Constraints
1	**34**	The only recorded cheating incident is on Sept. 19, 2022.
2	**33**	All recorded cheating incidents occurred in the year 2022.
3	**33**	No further constraints

Input 1

```
1995-09-30 00:30:00, 1999-05-28 23:59:59, 2004-05-15 09:12:26
```

Output 1

```
There have been 3 incidents of cheating in the past.
```

Input 2

```
1949-01-22 01:36:36
```

Output 2

```
There has been 1 incident of cheating in the past.
```

Problem C
Inverse Ferb Latin
Time Limit: 2 seconds

Invention of Ferb Latin *circa 2011, colorized*

Long ago, Phineas and Ferb created a new language and culture called "Ferb Latin." The rules were simple back then: One must take the first letter of every word, then move it to the end, then say "erb"; though two-letter words stay the same. They also added some non-verbal cues, such as reaching for the sky to say "okay" and cracking your knuckles to say that you're hungry. They then went all around the world to spread their new mode of communication

Recently, Phineas has observed that people are reverting back to the English language because they have become bored with the original Ferb Latin. "Boredom is something up with which I will not put!" he exclaims. And thus, Phineas and Ferb decide to modify their invented language.

They and their friends go to their backyard once again to formulate the rules for the modified Ferb Latin. Here is what they came up with:

1. If the word does not contain any vowels, insert `erb` after the first letter.
2. If the word starts with a vowel, simply put `ferbe` at the end.
3. If the word starts with a consonant but does contain a vowel somewhere, take every consonant before the first vowel and move that prefix to the end. Then finally, add `werb` to it at the end.

For the purposes of this problem, the letter `y` is **always** considered a consonant.

A few hours afterwards, Candace wakes up from her sleep and hears every person on their neighborhood talking differently. As always, she suspects Phineas and Ferb, and calls her friend Stacy. However, she finds that even Stacy is talking differently as well! She now asks you, her favorite programmer, for some immediate help.

Candace will give you words that she suspects are written in the modified Ferb Latin language, and your task is to help her decipher them back into plain English. While helping her, you notice that some of the given words are impossible to uniquely decipher back into English, and some aren't modified Ferb Latin at all! Whatever the case, please help Candace before she goes crazy.

For this problem, any non-empty string of lowercase letters counts as an English word.

Input Format

The first line of input contains T, the number of test cases. Each of the next T lines consists of a single word that must be translated back into English.

Output Format

Output the answer to each test case, each in its own line. For each word, the answer is determined as follows:

- If there are multiple possible English words which, in modified Ferb Latin, all translate to the given word, then output `SECRET`
- If there is *no* English word which, in modified Ferb Latin, translates to the given word, then output `NONE`
- Otherwise, output the unique English word (in all lowercase) which, in Ferb Latin, translates to the given word.

Constraints and Subtasks

For all subtasks
$1 \leq T \leq 500$ $1 \leq$ length of each word ≤ 100 Each word consists only of lowercase English letters

Subtask	Points	Constraints
1	**34**	length of each word ≤ 5
2	**33**	length of each word ≤ 7
3	**33**	No further constraints.

Sample I/O

Input 1	Output 1
3 iferbe serbky ifwerb	i sky fi

Explanation

In the sample input, each word Candace gave can be deciphered back to the English language:

- `i` is the only word that can be translated to `iferbe` using the second rule.
- `sky` is the only word that can be translated to `serbky` using the first rule
- `fi` is the only word that can be translated to `ifwerb` using the third rule

Problem D
Amogus??????????

Time Limit: 2 seconds

May isang manlilinlang sa'tin.

The video game Sa Atin, although released back in 2018, *exploded in popularity in* 2020, *becoming a cultural milestone that will surely be remembered for centuries to come. In fact, many sociologists posit that Sa Atin represents the peak of human achievement. All stories that follow will forever live in the shadow cast by this titan. And no game will ever be deemed perfect again, stained by the sin of simply* not being Sa Atin.

Not to mention the meme culture! Hahaha! Hilarious! Whenever I see any object with a passing resemblance to a Manlilinlang, I burst into uncontrollable laughter. Truly, this joke will ***never*** *get old!*

Today, we are going to hunt for Manlilinlang patterns in seemingly-arbitrary binary strings!

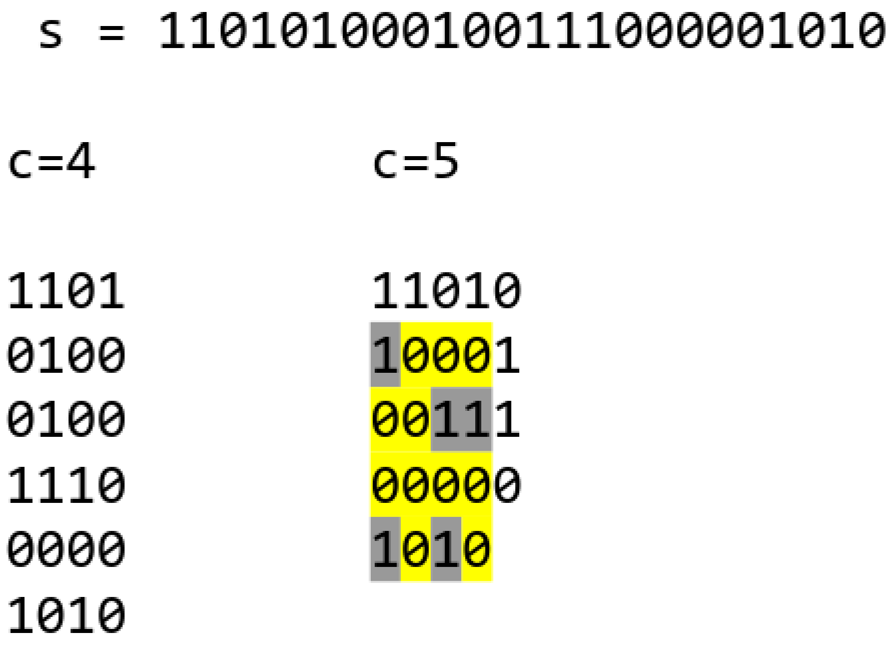
Suppose we have a 2D grid which contains only the characters `0` or `1`. A Manlilinlang pattern is either of the following 4×4 patterns appearing as a subgrid:

```
1000    0111
0011    1100
0000    1111
1010    0101
```

You will given a long binary string s (so its characters are only `0` or `1`). We are going to transform it into a grid by "word wrapping". Formally, conduct the following process:

- First, decide on a "column count" c.
- On the first row of the grid, place the first c characters of the string s.
- On the second row of the grid, place the next c characters of the string s
- On the next row, place the next c characters of the string s... and so on.
- Repeat this until we reach a row that has $\leq c$ characters.
  - If the last row has $< c$ characters, we left-align it.

The choice of c affects the resulting word-wrapped pattern. Consider the string `110101000100111000001010` (which has 24 characters). Here are the resulting word-wrapped grids for $c = 4$ and $c = 5$, for example. Note the incomplete final row for the $c = 5$ case.

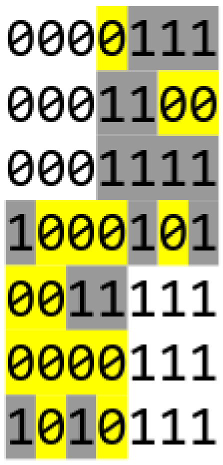


```
s = 110101000100111000001010

c=4        c=5

1101       11010
0100       10001
0100       00111
1110       00000
0000       1010
1010
```

Also note that a Manlilinlang pattern was found when $c = 5$, which we have highlighted. That's what we're looking for!

Given the string s, find the *maximum* possible number of Manlilinlang patterns in the word wrapped grid if the column count c is chosen optimally.

We also clarify that Manlilinlang patterns may overlap. For example, in the following grid, we count *two* Manlilinlang patterns.



```
0000111
0001100
0001111
1000101
0011111
0000111
1010111
```

In addition, rotations and reflections of the Manlilinlang pattern should **not** be counted; only count subgrids which appear exactly as described in the problem statement.

Also, a tip for those submitting in Python: Your code is usually significantly faster when it is submitted to PyPy instead of CPython.

Input Format

The first line of input contains a single integer, the length of the string s.

The second line of input contains the binary string s.

Output Format

Output a single integer, the maximum achievable number of Manlilinlang patterns.

Constraints and Subtasks

For all subtasks
s consists of only the characters `0` and `1`

Subtask	Points	Constraints
1	**51**	$\lvert s\rvert = 16$
2	**49**	$16 \le \lvert s\rvert \le 750$

Sample I/O

Input 1	Output 1

`16`
`1000001100001010`
	`1`

Input 2	Output 2

`24`
`110101000100111000001010`
	`1`

Input 3	Output 3
49 0000111000110000011111000101001111100001111010111	2

Problem E
Zoom Proctoring
Time Limit: 1 second

The National Olympiad for Intellectuals is a contest attended by high school students, consisting of a 3-hour exam to be answered individually. Due to logistical difficulties, the proctor responsible for overseeing the exam has to proctor the exam over Zoom.

No, not like that. I mean the proctor is unable to make it onsite, but the students are, so the proctor has to observe the classroom over Zoom. The proctor has an ultrawide monitor at home though, so to maximize her proctoring abilities, she asks the students to arrange the chairs to be in one long line to make the most of her ultrawide monitor.

There are n students participating in the contest, numbered from 1 to n, with each student hailing from one of m different schools. The classroom has n chairs arranged in a single line, and each student must be seated in one of the n chairs. To prevent students from committing *intellectual* dishonesty, the proctor wishes to have a *safe* seating arrangement. A seating arrangement is considered *safe* if and only if no two adjacent chairs are occupied by students hailing from the same school.

Write a program which generates, if possible, a safe seating arrangement for the n students.

Input Format

The first line of input contains two space-separated integers n and m, the number of students and the number of schools, respectively.

The second line of input contains n space-separated names. The ith of these names denotes the school of the student whose number is i.

Output Format

If there is no safe seating arrangement, output a single line containing `IMPOSSIBLE`.

Otherwise, if there is a safe seating arrangement, print a single line containing `POSSIBLE`, followed by a line containing n space-separated integers, the student numbers corresponding to a safe seating arrangement. If there are multiple possible safe seating arrangements, you may output any of them.

We remind you to make sure that all lines in your output end with a newline `\n` character.

Constraints and Subtasks

For all subtasks
$2 \leq n \leq 10^5$
$2 \leq m \leq n$
A school name is at most 10 characters long.
Each school name only consists of lowercase English letters.
There are exactly m different names of schools in the input.

Subtask	Points	Constraints
1	**20**	$2 \leq n \leq 10$
2	**30**	$m = 2$
2	**40**	$m = 3$
3	**10**	No further constraints

Input 1

```
5 3
admu up up dlsu admu
```

Output 1

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

Input 2

```
8 4
up ust dlsu up admu up up up
```

Output 2

```
IMPOSSIBLE
```

Problem F
Rule of Three

Time Limit: 2 seconds

Your programming teacher in HS was a very wise woman. She would dispense invaluable life lessons that her students would take to heart. There are three core pieces of advice which her students would affectionately dub the "`new`* rules".*

1. Don't pick up the phone. You know he's only calling 'cause he's drunk and alone.
2. Don't let him in. You'll have to kick him out again.
3. In C++, if the destructor, copy constructor, or copy assignment operator are defined for a class, then **all three** should be defined for that class.

In 2011*, two more rules were added to the* `new` *rules, giving a total of five.*

4. The move constructor and move assignment operator should both also be included in the methods listed in Rule 3 when using C++11 or greater.
5. **Only buy items that cost a whole number of pesos.**

Huh, that last one seems weirdly specific. But it's the only one that is immediately actionable, so let's try practicing it right now!

On the last day of school, your programming teacher set up the following exhibit. She has n^3 items for sale, which she has arranged into a uniform $n \times n \times n$ cube. She writes n numbers $a_1, a_2, \dots, a_n$ along its length. She similarly writes the n numbers $b_1, b_2, \dots, b_n$ along the width, and the n numbers $c_1, c_2, \dots, c_n$ along the height. Using the *Rule of Three*, she says that the object in the ith row from the top, jth column from the left, and kth level from ground is being sold at a price of $a_i + b_j + c_k$ **centavos**.

Remembering the fifth of her `new` rules, you realize that she is testing you and wants you to only buy the items that cost a whole number of pesos. So, how many of these n^3 items cost a whole number of pesos, if their price is computed using the Rule of Three?

You are reminded that there are 100 centavos to a peso.

Also, a tip for those submitting in Python: Your code is usually significantly faster when it is submitted to PyPy instead of CPython.

Input Format

The first line of input contains a single integer n.

The second line of input contains the n space-separated integers $a_1, a_2, \dots, a_n$.

The third line of input contains the n space-separated integers $b_1, b_2, \dots, b_n$.

The fourth line of input contains the n space-separated integers $c_1, c_2, \dots, c_n$.

Output Format

Output a single integer, the number of items which are worth a whole number of pesos.

Constraints and Subtasks

For all subtasks
$1 \le n \le 333\ 333$ $1 \le a_i, b_i, c_i \le 3\ 333\ 333\ 333$ for all $1 \le i \le n$

Subtask	**Points**	**Constraints**
1	**20**	$1 \le n \le 333$
2	**27**	$1 \le n \le 3\ 333$
3	**23**	$1 \le a_i, b_i, c_i \le 34$ for all $1 \le i \le n$
4	**30**	No further constraints

Input 1	**Output 1**

```
3
3 33 333
555 666 777
12 1001001 2022
```
	`4`

Explanation

In the sample input, the items at these coordinates are worth a whole number of pesos:

- $i = 2$ and $j = 2$ and $k = 2$, worth exactly 10017 pesos.
- $i = 3$ and $j = 2$ and $k = 2$, worth exactly 10020 pesos.
- $i = 2$ and $j = 1$ and $k = 1$, worth exactly 6 pesos.
- $i = 3$ and $j = 1$ and $k = 1$, worth exactly 9 pesos.

Problem G
I Really Hate Olympiad Algebra

Time Limit: 3 seconds

Princess Mattea Oleifera is the uncontested most popular girl at Olympia High, due to her supreme intelligence as well as her family's history of successes at Olympia High throughout the generations. But a transfer student arrives, hungry to prove himself and show everyone what he can do. Neumann Philips is ready to take the world by storm!

On his first day of school, Neumann challenged Mattea to a duel for the spot of Smartest Student at Olympia High. Mattea answered that she was flattered, but that Neumann had to prove himself worthy before she would accept such a challenge. For instance, if he was so smart, then surely a puzzle like this wouldn't be so hard, would it?

We say that some positive integer n is *arbitrarily-k-interesting* if there exist positive integers x, y, and z such that

$$n = xyz + 4xy + 2xz + yz + 8x + 4y + 2z + k.$$

Yuck! What a horrid expression! But when Neumann said this, Mattea just smirked and said that oh it really wasn't *that* bad.

Given positive integers k, L, and R, all Neumann has to do is count the number of arbitrarily-k-interesting integers in the range L to R (inclusive). Formally, count the number of integers n such that $L \leq n \leq R$ and n is arbitrarily-k-interesting.

Input Format

Input consists of a single line containing the three space-separated integers k, L, and R.

Output Format

Output a single integer, the number of arbitrarily-k-interesting integers in the given range.

Constraints and Subtasks

For all subtasks
$1 \leq k \leq 10^{18}$

Subtask	Points	Constraints
1	**7**	$1 \le L \le R \le 10$
2	**18**	$1 \le L \le R \le 50$
3	**20**	$1 \le L \le R \le 1000$
4	**15**	$1 \le L \le R \le 10^6$
5	**20**	$1 \le L \le R \le 5 \times 10^7$
6	**13**	$1 \le L \le R \le 10^8$
7	**6**	$1 \le L \le R \le 10^{10}$
8	**1**	$1 \le L \le R \le 10^{12}$

Input 1	Output 1
`2 20 40`	`5`

Explanation

For example, we can show that 39 is arbitrarily-2-interesting, because $39 = xyz + 4xy + 2xz + yz + 8x + 4y + 2z + 2$ if we choose $x = 2$, $y = 1$, and $z = 1$.

Problem H
The World Is Round

Time Limit: 2 seconds

You may have heard of the Flat Earth Society that believes that the Earth is flat. But everyone already knows they're wrong.

The Earth is actually a ***circle****.*

In the beginning of last Thursday, a Geologically Omnipotent Deity created the circular world. Initially it was without form, and void. So the Deity thought that the world would be way cooler if it had some mountains!

Let's divide the Earth into n sections, labeled 1, 2, 3, $\ldots$, n in the clockwise direction. Actually, the world is a circle, so after the nth section comes the $(n+1)$th section, which is the same as the 1st section (but just given a different name); similarly, after *that* comes the $(n+2)$th section, which is the same as the 2nd section, and so on. Also, before the 1st section comes the 0th section, which is the same as the nth section (but just given a different name); similarly, before *that* comes the (-1)th section, which is the same as the $(n-1)$th section, and so on.

Initially, all sections had a *height value* of 0 (because the world was locally flat). Then, the Deity created the mountains by executing the following process q times:

- Choose a section v and a value k (this choice is independent of all the choices that came before it).
- Update the height values as follows:
  - Increase the height of the vth section by k.
  - Increase the height of the $(v-1)th$ section by $k-1$, and then increase the height of the $(v+1)$th section by $k-1$.
  - Increase the height of the $(v-2)$th section by $k-2$, and then increase the height of the $(v+2)$th section by $k-2$
  - $\vdots$
  - Repeat this until k is 0.
- We emphasize that it may be possible for e.g. the $(v-2)$th and the $(v+2)$th sections to both refer to the *same* section, and in this case, that section gets its height increased twice by that step.

And the Deity saw everything that it had made, and... well it was okay, it thought. It's certainly done better in the past, but this was passable enough.

What was the height of each section in the circular world after all q updates? These values can get quite large, so you only need to output the remainder of each height it is divided by $10^9 + 7$.

Oh also, we hope you don't mind answering T independent test cases.

Input Format

Input begins with a single integer T, the number of test cases.

Each test case begins with a line containing two space-separated integers n and q. This is followed by q lines that each contain two space-separated integers v and k, the values chosen for that particular update.

Output Format

For each test case, output a line containing n space-separated integers, the final height values of sections 1 through n (in that order). Again, these values can be quite large, so you only need to output the remainder when each one is divided by $10^9 + 7$.

Constraints and Subtasks

Let N be the sum of n across all test cases, and let Q be the sum of q across all test cases.

For all subtasks
$1 \le T \le 3 \times 10^5$
$1 \le N \le 3 \times 10^5$
$1 \le Q \le 3 \times 10^5$
$1 \le v \le n$ and $1 \le k \le 10^{18}$ for each update.
$1 \le n, q$ in each test case.

Subtask	Points	Constraints
1	**20**	$T \le 100$ $n \le 100$ in each test case $q \le 100$ in each test case $k \le 100$ for each update
2	**20**	$q \le 1000$ in each test case $Q \le 3000$ $k \le 2000$ for each update
3	**15**	$N \le 10^5$ $k \le \lceil \frac{n}{2} \rceil$ for each update, in each test case
4	**15**	$k \le n$ at each step, in each test case
5	**17**	$n, q \le 1000$ in each test case $N \le 3000$ $Q \le 3000$
6	**13**	No further constraints.

Input 1

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

Output 1

```
4 3 3 3 4
```

Input 2

```
1
2 1
1 2
```

Output 2

```
2 2
```

Explanation

In the first sample input, here are the effects of the first update:

- Increase the 3^{rd} section's height by 3.
- Increase the 2^{nd} and 4^{th} sections' heights by 2.
- Increase the 1^{st} and 5^{th} sections' heights by 1.

This will result in the heights of $\{1, 2, 3, 2, 1\}$ for the sections. After the second update, the heights become $\{3, 3, 3, 2, 2\}$, and after the third update, the heights become $\{4, 3, 3, 3, 4\}$.

For the second sample, here are the effects of the first (and only) update:

- Increase 1^{st} section's height by 2.
- Increase 0^{th} and 2^{nd} sections' heights by 1.

Note that "the 0^{th} section" and "the 2^{nd} section" are different names for the same thing; thus, this section's height is increased by 1 **twice**.

Problem I
Who Doesn't Love Sliding Puzzles?

Time Limit: 1 second

Man, this sure is a nice video game. The controls are smooth, the movements are responsive, the core gameplay loop is quite fun... It sure would be quite a shame if we were to ruin that with an inane minigame. A ***non-optional*** *minigame! And you need to revisit it and beat it on the highest difficulty if you want to 100% this game!*

The playing area has $n+3$ rune sigils, labeled 1 to $n+3$. For each pair of sigils u and v such that $1 \le u < v \le n+1$, at least one of the following is true: there is a magic arrow from u to v, or there is a magic arrow from v to u, or both. This is also true for the pairs of sigils $(n+1, n+2)$, and $(n+2, n+3)$, and $(n+3, n+1)$ (a magic arrow exists connecting one sigil in each pair to the other one, or vice versa, or both).

Sigils 1 to n each contain a stone. Each stone is painted with a label from 1 to n, and each value is distinct. Note that sigils $n+1$, $n+2$, and $n+3$ initially **do not** have a stone in them.

Our goal is to make each stone match the label of the sigil it's on. In order to do this, we must shuffle the stones around. But we must respect the flow of magical energy and also make sure no sigil is overloaded; thus, this is your only operation:

- Choose a stone on some sigil u, and another sigil v such that there is **no** stone on v and a magic arrow exists from u to v; then, move the stone from sigil u to v (now sigil u is empty and sigil v has that stone).

Now, this is quite hard, so you are allowed up to **two cheats** at the start of the game. For each cheat, you select two sigils u and v such that there is a magical arrow from u to v but not vice versa; then, add that magical arrow that goes from v to u.

Now this is a perfectly fun and quite interesting brain teaser, but unfortunately we just want to go back to the main gameplay. Can you write a program that plays this game for us instead?

If the game is unsolvable, please say so as well.

Input Format

The first line of input contains a single integer n.

The second line of input contains n space-separated integers, encoding the original positions of the tokens. Recall that each of the sigils from 1 to n initially begins with a stone on it; the label of the stone on the ith sigil is equal to the ith integer in this line.

Then, $n+3$ lines follow, each containing a string of $n+3$ characters, all `0` or `1`. This encodes the magic arrows. For the ith line, if the jth character is `1`, then there is a magic arrow from sigil i to sigil j; otherwise, no such magic arrow exists.

Output Format

Output a line containing the word `YES` if the task is possible, and `NO` if not.

If `YES`, then you must also construct a solution.

Next, output a line containing two space-separated integers R and M, where R is the number of cheats you will use, and M is the number of moves to be performed. Here, $0 \le R \le 2$ and $0 \le M \le 4 \times 10^5$ must both hold.

Next, output R lines, each describing a cheat. Each line should contain two space-separated integers u and v, meaning that you wish to draw a magic arrow from sigil u to v or v to u for this cheat (whichever one is missing).

Next, output M lines, each containing two space-separated integers u and v, meaning that you wish to move the stone from sigil u to sigil v.

If any of the cheats or moves are invalid for reasons described in the main problem statement, you will be marked incorrect.

Any solution will be accepted so long as the stones are properly arranged after all moves have been executed. As a reminder, you can only perform at most 4×10^5 moves. We can show that if a solution exists, then one exists with an M in this range.

Constraints and Subtasks

For all subtasks
$1 \le n \le 50$ It is guaranteed that the given magic arrows satisfy the constraints described in the problem statement.

Subtask	Points	Constraints
1	**20**	$n = 3$
2	**20**	$n \leq 4$
3	**10**	If there is a magic arrow from sigil u to v, then there is also one from sigil v to u.
4	**35**	For all u, v such that $1 \leq u < v \leq n+1$, there exists a magic arrow from u to v. This also holds true for $(u, v) \in \{(n+1, n+2), (n+2, n+3), (n+1, n+3)\}$.
5	**9**	$n \leq 25$
6	**6**	No further constraints

Input 1	Output 1

```
3
3 1 2
010000
000100
110100
110011
000101
000110
```
	```
	YES
	1 6
	2 3
	2 4
	1 2
	4 1
	3 4
	2 3
	4 2
	```

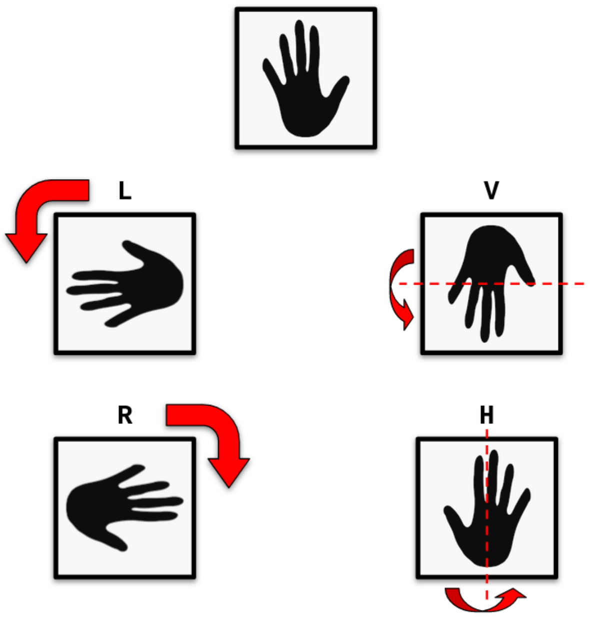
Problem J
.img of the Artist as Filipino

Time Limit: 1 second

Bob enjoys the stained glass mosaics that he had learned to create last December. However, while it is fun to play with arts and crafts, his task for today is to train an arguably more practical skill in the 21st century—digital art.

Bob has digitized his mosaic into a bitmap, representing it as a $n \times n$ grid. Each pixel is colored either black or white, which we encode with the characters `0` and `1` respectively. Each pixel is completely indistinguishable from any other pixel of the same color, even if you rotate the image or flip it over.

Cindy, meanwhile, has been practicing how to use programming in order to manipulate images. She is once again enthralled by Bob's mosaic, and wants to apply some sequence of transformations to it. She has four main types of operations which she wishes to perform, which we show below.



We describe (and encode) these operations as follows.

- `L` to rotate the mosaic 90° counter-clockwise
- `R` to rotate the mosaic 90° clockwise
- `H` to reflect the mosaic horizontally (across its vertical axis)
- `V` to reflect the mosaic vertically (across its horizontal axis)

A *command string* is a string which consists only of the four letters `L`, `R`, `H`, and `V`. Cindy wrote a program such that, when fed a command string, it iterates over each character in the string from left to right, and then applies the corresponding transformations to Bob's mosaic, in order. The resulting image is printed out, and then **the mosaic is restored to its original state.**

Alice hears about these two digital creations and thinks of a fun puzzle! She first produces a long "reference string" s which consists of m characters, all of which are `L`, `R`, `H`, or `V`. This reference string is 1-indexed, i.e. its first character is s_1 and its last character is s_m. She wants to try writing a program that can handle Q queries, each being one of two types:

- `? i j`. Consider the substring $s_i s_{i+1} \dots s_j$. If *only this substring* is taken as a command string and fed into Cindy's program, what is the output?
  - Printing out the entire grid would take too much time, so it will be sufficient for you to only output its **hash** value (the hash function is described in the Output Format).
- `! i j p` where $p = p_1 p_2 p_3 p_4$ is a permutation of `LRHV`. *Relabel* the characters $s_i, s_{i+1}, \dots s_j$ in the reference string. These are all done simultaneously:
  - Replace all `L` with p_1
  - Replace all `R` with p_2
  - Replace all `H` with p_3
  - Replace all `V` with p_4

Alice easily solved this problem, but she wants to double check that her program is correct. You don't mind being her tester, do you?

Input Format

The first line of input contains a single integer n.

Then, n lines follow, each containing a string of length n. This encodes Bob's $n \times n$ mosaic. The colors black and white are represented by `0` and `1` characters respectively.

The next line contains the integer m, the length of the reference string s.

The next line contains the reference string s itself.

The next line contains a single integer Q, the number of queries that your program must process.

Then, Q lines follow, each describing either a `?` or `!` query in the format given in the problem statement.

Output Format

For each ? query, output a line containing the hash of the mosaic after using the indicated substring as a command string.

Recall that the mosaic is a grid consisting only of the characters `0` and `1`. Its hash is computed as follows:

1. "Flatten" the grid, i.e. concatenate each of its rows into one very long string, in order from top to bottom.
2. Interpret the resulting string as a binary number, and output its value in base 10, modulo 694201337 (a *totally arbitrarily chosen* prime number).

For example, consider the following grid.

```
010
010
110
```

To get its hash value, we first flatten it into the string `010010110`, which we interpret as a binary representation for the number 150. Finally, 150 mod 694201337 is still equal to 150.

Constraints and Subtasks

For all subtasks
$3 \le n \le 750$ $1 \le m \le 2 \times 10^5$ $1 \le Q \le 2 \times 10^5$ For all queries, $1 \le i \le j \le m$ There always exists at least one ? query.

Subtask	**Points**	**Constraints**
1	**30**	$n \le 3$ $m \le 750$ $Q \le 750$
2	**20**	There are no ! queries.
3	**20**	In all ! queries, $i = j$.
4	**8**	`L` and `R` do not appear in s. In all ! queries, $p_1 =$ `L` and $p_2 =$ `R`
5	**15**	In all ! queries, $i = 1$ and $j = m$.
6	**7**	No further constraints.

Input 1

```
3
110
010
010
7
LRRHVVL
5
? 1 7
? 2 5
! 3 7 RVHL
? 1 7
? 2 5
```

Output 1

```
150
402
120
147
```

Explanation

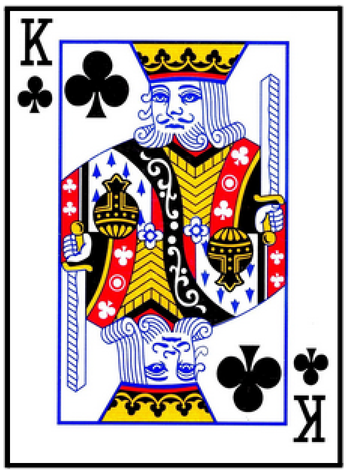
Remember that the mosaic is restored to its original state after each `?` query.

In the sample input, the query `! 3 7 RVHL` transforms s from `LRRHVVL` to `LRVHLLR`.

Problem K
Osmosis (Version Alpha)

Time Limit: 4 seconds

【DIFFICULTY LEVEL】



【GAME】
「Osmosis」

【RULES】

See below for current version of rules.
***TODO**: Gather feedback from playtesters*

All play takes place on a grid with r rows and c columns. The objective of this game is for the players to navigate through a maze of obstacles and earn points by collecting treasure!

This game is played with a team of x players. All players start on a special cell called the base. In 1 second, a player can move from their current cell to any of the four cells directly to its north, south, east, or west, provided that there are no obstacles in that cell and that they never exit the grid. They may also opt to rest, and spend 1 second without moving from their current cell. Diagonal movement is not allowed.

All players move simultaneously, and it is permissible for multiple players to occupy the same cell (after all, everyone starts at the base). The game ends after T seconds have passed.

Exactly n of the cells on this grid contain treasure! The first time that any player enters a cell that contains treasure, that treasure is collected and their team is awarded a fixed number of points (but this value may be different per treasure). We reiterate that each treasure's points can only be collected **once** across the whole team.

Additionally, there is one special cell that contains the *enemy* base. The enemy base also has a once-collectible fixed number of points, but it requires that *all x players* be there at the same time in order for these points to be obtained.

Formally, each treasure's points are obtained if and only if there exists some time $t \le T$ such that *any* of the players are standing at the cell containing that treasure at that time t. The points from the enemy base are obtained if and only if there exists a time $t \le T$ such that *all* x players are standing at the enemy base at that time t.

Given the layout of the grid, help us determine the answer to the following question: What is the maximum number of points that the players can acquire within the time limit?

Input Format

The first line of input contains these six space-separated integers:

- r the number of rows
- c the number of columns
- n the number of cells with treasure
- x the number of players
- T the time limit of the game
- the points earned if all x players are on the enemy base

This is followed by a line containing n space-separated integers, the values of the treasures. The ith of these integers denotes the number of points awarded for collecting the ith treasure.

This is followed by r lines that contain c characters each, denoting the grid. The meaning of each of the characters in these lines is as follows:

- $\star$ denotes the players' base
- ! denotes the enemy's base
- . denotes a passable cell
- # denotes a cell with an impassable obstacle
- A capital English letter denotes a cell with treasure.
  - The ith treasure is denoted by the ith letter of the alphabet.

- So, for example, the cell marked `A` corresponds to the first treasure, and so collecting it will award a number of points equal to the 1st integer in the second line of input.

Output Format

Output the maximum number of points that can be obtained, assuming optimal play.

Constraints and Subtasks

For all subtasks
$1 \le r, c \le 1000$ $1 \le n \le 15$ $1 \le x \le 5$ $1 \le T \le 10^9$ $1 \le$ All point values $\le 10^9$

Subtask	**Points**	**Constraints**
1	**10**	$n = 1$
2	**5**	$n = 2$
3	**10**	$x = 1$ $n \le 8$
4	**15**	$x = 1$ $T \le 200$
5	**5**	$x = 1$
6	**30**	$x = 2$
7	**25**	No further constraints

Input 1	**Output 1**

`3 5 2 1 8 1500`
`900 1100`
`*#!#A`
`.#.#B`
`.....`
	`2000`

Input 2

```
3 5 2 1 8 2500
900 1100
⋆#!#A
.#.#B
.....
```

Output 2

```
2500
```

Input 3

```
3 5 2 2 8 1500
900 1100
⋆#A#B
.#.#.
..!..
```

Output 3

```
3500
```

Input 3

```
3 5 2 2 8 1500
900 1100
⋆#A#B
.#.#.
....!
```

Output 3

```
2600
```

Explanation

In the first two sample inputs, there is only $x = 1$ player.

- In the first sample input, we collect both treasures for $1100 + 900 = 2000$ points.
- In the second sample input, it is instead more optimal to capture the enemy base for 2500 points.

In the next two sample inputs, there are $x = 2$ players.

- In the third sample input, both players first capture the enemy base together, and then they split up to get both treasures within the time limit, for $1500 + 900 + 1100 = 3500$ points.
- In the fourth sample input, it is again optimal to capture the base. But this task requires both players to be present, and so there isn't enough time to go back and capture the first treasure (marked `A`) within the time limit. This earns us $1500 + 1100 = 2600$ points.

Problem L
Accomplice to the Cutest Bunny in the World

Time Limit: 4 seconds

Oreo McFluffy, the unrepentant.

Oreo, the cutest bunny in the entire world, has decided to do some redecorating, as rabbits are wont to do. He decided to tear up the paint on the walls and now there are holes everywhere!

Let's model your wall using the non-negative quadrant of the 2D plane. The bottom-left corner represents the origin $(0, 0)$, and (x, y) refers to the point that is x units east and y units north of the origin. Your wall is M units wide, but for some reason, arbitrarily tall.

Oreo created n holes on the wall, which we model as points; the ith point is located at (x_i, y_i). In addition, the baseboards have been torn up, so we also need to hide **all** points $(x, 0)$ with $0 \leq x < M$. Our foolproof plan is to cover up all the damage by using some tasteful paintings by local artists.

A painting is, for all intents and purposes, just a rectangle (sorry artists) that can be placed on the wall. A painting covers a damaged point on the wall if that point lies on or within its rectangular shape.

Each square unit of painting costs 1 peso (we're *really* sorry, artists), so for example an $L \times W$ rectangular painting would cost LW pesos. The artists are even skilled enough to create one-dimensional paintings (e.g. a $0 \times W$ rectangular painting), which are basically just line segments, and they cost 0 pesos (once again, we're really sorry to all the artists).

No two paintings may overlap, other than possibly at their edges or corners. As a special exception, because of their weirdness, any one-dimensional painting cannot overlap with other paintings, except possibly at its endpoints.

Furthermore, the **bottom-left** corner of each painting must rest on the ground (i.e. have y coordinate zero), while the **top-left** corner of each painting must be placed on a *hook*. Note that for the one-dimensional paintings which are horizontal line segments, the top-left and bottom-left corners are one and the same.

There are h possible locations on the wall where a hook can be placed. The jth candidate location is at (p_j, q_j), and it costs some c_j pesos in order to buy a hook and place it there. There is also a special hook at $(0, 0)$; it costs 0 pesos to use since there is already a hook there.

Find the minimum cost needed in order to masterfully hide the consequences of Oreo's innocent vandalism. Additionally, construct such a solution—we can show that the hooks placed on the wall actually uniquely determine the paintings that hide all the damage (if valid), so just provide that set of hooks.

Of course, it's also possible that this task is impossible, and in that case just say so as well.

Input Format

The first line contains the three space-separated integers n, h, and M.

The next n lines each describe a hole in the wall. The ith line contains two space-separated integers (x_i, y_i), the coordinates of the ith hole.

The next h lines each describe a candidate hook location. The jth line contains two space-separated integers (p_j, q_j), the coordinates of the jth hook location.

The last line contains h space-separated integers $c_1, c_2, \ldots, c_h$, where c_j is the cost of using the jth candidate hook location from the input.

Output Format

If there is no way to use the hooks to satisfy the conditions, output one line containing just the number `-1`

Otherwise, output one line containing the minimum cost to hide all the damage.

Then, let h_s be the number of hooks you need to buy for your solution. Output the value of h_s in its own line. Then, output a line containing h_s space-separted integers, the indices of the hooks your solution uses (1-indexed) in any order.

There is no need to output the hook at $(0, 0)$ if you want to use it, since it is always already there. (In fact, since $(0, 0)$ is not given an index in the input, there is no way to even output it.)

Constraints and Subtasks

For all subtasks
$0 < x_i, p_i < M \le 10^9$
$1 \le y_i, q_i \le 10^9$
$0 \le c_i \le 10^{12}$
$1 \le n \le 5 \times 10^5$
$1 \le h \le 5 \times 10^5$
No two hooks share the same location.
No two points share the same location.

Subtask	Points	Constraints
1	**14**	$h \le 10$ $n \le 100$
2	**20**	$h \le 1000$
3	**11**	$n = 1$ $c_j = 0$ for all j
4	**20**	$n = 1$
5	**24**	$n, h \le 100000$
6	**11**	No further constraints.

Input 1

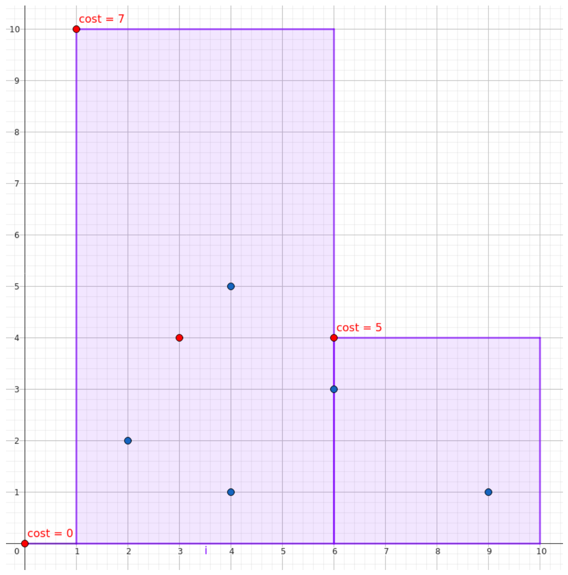
```
5 3 10
2 2
4 5
4 1
6 3
9 1
1 10
3 4
6 4
7 12 5
```

Output 1

```
78
2
3 1
```

Explanation

Below is an illustration of the solution in the sample output:



We have

- a painting of height 10 and width 5 placed on hook 1, which costs a total of $5 \cdot 10 + 7 = 57$ pesos, and
- a painting of height 4 and width 4 placed on hook 3, which costs a total of $4 \cdot 4 + 5 = 21$ pesos.

In total, this uses 78 pesos. It can be shown that this solution is optimal.

Problem M
Pre-In-Post

Time Limit: 2 seconds

Curious Jorge is curious about many things, and that includes his family's lineage! He spent all weekend researching on the sprawling family tree of his ancestors. He worked so hard on it that I sure hope nothing bad happens to it!

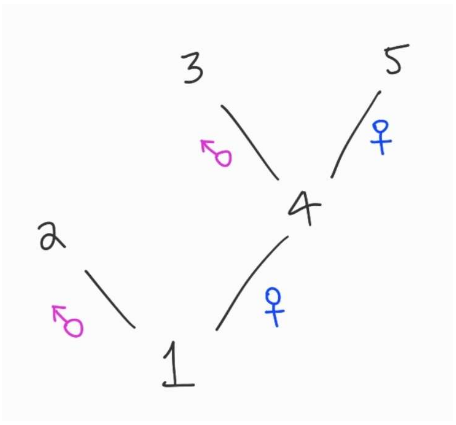
Curious Jorge defines a family tree as follows.

- A family tree consists of a collection of n different people that are connected by parent-child relationships.
    - In lieu of names, we instead give each person (including Curious Jorge) a unique integer label from 1 to n.
- Every person on the tree is either Curious Jorge himself (the "root") or the mother or father of someone else in the tree.
- For one reason or another, a person on the tree might be missing information on their mother, or missing information on their father, or both.

Curious Jorge hopes that you can see that given n, there are only finitely many family trees with n people (although this number can be **huge**). And thank the heavens that this is true, because Curious Jorge accidentally destroyed the structure of his tree. Oh no!

Curious Jorge lost all information on the parent-child relationships. Furthermore, he forgot whom each label corresponds to, so he doesn't know if some label corresponds to someone's mother or someone's father or *himself*, so any label can go anywhere in the tree.

The *only* thing that Curious Jorge remembers about his family tree is that it is *Pre-In-Post-orderly*. In order to explain what that means, consider the following family tree. Each person's father (if known) is drawn on their left, and each person's mother (if known) is drawn on their right.



We can produce a permutation of the integers from 1 to n by conducting a *tree traversal*. In a tree traversal, we start at the "root" (the person with no parents), then recursively traverse through their father's family tree (if the father is known), then recursively traverse through the mother's family tree (if the mother is known).

We will print out the label of each person we visit. But each person can come in one of three possible modes...

- `pre` — we print their label before traversing either of their parents' family trees.
- `in` — we print their label after traversing their father's family tree, but *before* traversing their mother's family tree.
- `post` — we print their label after traversing both of their parents' family trees.

Curious Jorge is free to choose whatever mode he wants for each person, and each choice is made independently of the others. For example, in the above tree...

- He can choose `pre` mode for everyone, which prints out the labels in the order `12435`.
- He can choose `in` mode for everyone, which prints out the labels in the order `21345`.
- He can choose `post` mode for everyone, which prints out the labels in the order `23541`.
- He can choose `pre` for those with odd labels and `in` for those with even labels, which prints out the labels in the order `12345`.

A family tree is *Pre-In-Post-orderly* if it is possible to find any assignment of modes for each person such that when the family tree is traversed, the labels we print out are, in order, 1 to n.

Please help Curious Jorge! How many *different* Pre-In-Post-orderly family trees are there with n people? This can be quite huge, so only output the answer modulo a given prime p.

Two family trees are considered different if there exists a person such that any of the following are true:

- They have a mother in one tree but not the other
- They have a father in one tree but not the other
- Their mother is different between the two trees
- Their father is different between the two trees

Input Format

The first line of input contains only the integer n.

The second line of input contains only the integer p.

Output Format

Output a single integer, the number (modulo p) of Pre-In-Post-orderly family trees with n people.

Constraints and Subtasks

For all subtasks
$1 \leq n \leq 10^6$ $2 \leq p \leq 10^9$ and p is prime.

Subtask	Points	Constraints
1	**10**	$n \leq 5$
2	**7**	$p = 2$
3	**15**	$n \leq 15$
4	**15**	$n \leq 25$
5	**32**	$n \leq 1000$
6	**12**	$n \leq 50000$
7	**9**	No further constraints.

Input 1	Output 1
2 5	4