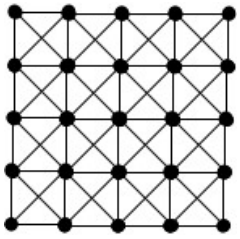
A — Networks

Given three "equidistant orthogonal networks" of dimensions $n \times n$, $m \times m$, and $p \times p$ (see the figure below for a 5×5 network; all horizontal and vertical edges are of 1 unit length; all diagonal edges are of $\sqrt{2}$ length), we are interested in the following three subnetworks:

- a minimal-length subnetwork that links (directly or indirectly) all the nodes of the network of size $n \times n$;
- a subnetwork for accessing from the top left node all the other nodes of the network of size $m \times m$ in minimal number of steps (or edges);
- a subnetwork for accessing from the bottom right node all the other nodes of the network of size $p \times p$ on minimal-length routes.

Only the number of edges of each subnetwork has to be printed to the standard output.



Input Specification

The input contains three numbers: n, m, and p, separated by a space ($2 \leq n, m, p \leq 10\,000$).

Output Specification

Print three numbers to the standard output, representing the number of edges for each network (separated by a space).

Sample Input

```
1. 2019 539 9876
```

Output for Sample Input

```
1. 4076360 290520 97535375
```

B — Open Data About Train Timetable

The National Railway Company of Landia decided to publish the whole train timetable information as open data. Being very enthusiastic, the web developer Andrew creates a website to include these open data. Bad surprise: the published information is just ordinary fake. Andrew analyzes very carefully the open data and discovers a lot of inconsistencies. The published information includes some fictitious trains, some trains are missing, the information about running restrictions are erroneous, etc.

Your task is to find all mismatches in a given timetable.

Input Specification

The input contains the following information:

- the first line contains the number of stations ($\leq 40\,000$) and the number of trains ($\leq 160\,000$);
- for each station: code (31-bit positive integer) and name (≤ 27 characters, including spaces);
- for each train: code, restriction (0: runs every day, 1: runs with restrictions), departure station code, departure time, arrival station code, arrival time (you don't need to use the train code, the departure time, and the arrival time).

Output Specification

You have to print the list of all mismatches (regarding the number of trains and restrictions). Mismatches must be lexicographically ordered by the name of the departure and arrival stations. Follow the format of the sample output.

Sample Input

```
4 9
15 Paris Est
11 Roma Ovest
12 Bucuresti Nord
13 Budapest Keleti
101 0 15 7:00 11 18:00
102 0 11 8:00 15 20:00
103 1 15 9:00 11 21:00
201 0 12 6:00 13 23:00
202 1 13 5:00 12 22:00
301 0 11 6:00 13 23:30
302 0 13 5:00 11 22:30
203 0 12 12:00 13 10:00
204 0 13 15:00 12 12:00
```

Output for Sample Input

```
12 Bucuresti Nord - 13 Budapest Keleti, restrictions
15 Paris Est - 11 Roma Ovest, number of trains
```

Explanation

There are two trains from Bucuresti to Budapest and two trains from Budapest to Bucuresti, but there is a mismatch regarding the restrictions.

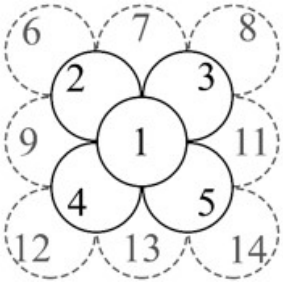
There are two trains from Paris to Roma and one train from Roma to Paris, therefore there is a mismatch regarding the number of trains. In this case, the other type of mismatch (regarding the restrictions) is not signaled.

Note that each route with mismatches must be considered only once. For example, signal only Paris – Roma, don't signal Roma – Paris.

C — Champagne Pyramid

After the successful preparation of the ECN contest, the organizers decided to open a bottle of champagne. One of the organizers came up with the idea to build a three-dimensional pyramid from the drinking glasses (see the figure above) and fill the glass on the top of the pyramid until every glass is full of champagne. Interestingly, all the glasses look the same, but they have different volumes. Constructing the pyramid, there are the following rules:

- The levels of the pyramid are numbered from 1 (on the top) to n (on the bottom).
- Except for the lowest (nth) level, every glass is placed on exactly 4 glasses below it.
- On the ith level, there are $i \times i$ glasses.
- The glasses are numbered following the rule on the following figure:



Once the gigantic pyramid is built, the organizers quickly open the bottle and start filling the top glass. The filling speed is exactly 1 drop per second. As a glass is filled, something funny happens. The first drop that hits the filled glass goes on to the lower level into the glass to the northwest (NW), the second drop to the northeast (NE), the third to the southeast (SE), and the fourth to the southwest (SW). The rule applies further on: the fifth drop goes on to the NW, the sixth to NE, the seventh to SE, the eighth to SW, etc. For example, in the figure above, when the 1st glass becomes full, the drops will go on into the 2nd, 3rd, 5th, and 4th glass in this order, concentrically. Another example, when the 4th glass becomes full, the drops go on concentrically to the 9th, 10th, 13th, and 12th glass.

It has also been observed by the organizers that some of the glasses at the lowest level are filled in some cases sooner than the others, and some drops of champagne spill on the table.

Knowing that the organizers don't want to waste the champagne on the table, they thought they would first calculate the number of wasted drops of champagne, and only then decide whether to make this spectacular pyramid or not. The organizers are tired, and they ask you to write a program for them that, knowing the number of levels n and the volume of each glass V_i, calculates the minimum number of seconds needed for all the glasses to become full and the number of wasted drops at that moment.

Input Specification

The first line of the input contains one integer n, which is the number of levels of the pyramid ($1 \le n \le 25$).

The next line contains $m = n(n + 1)(2n + 1) / 6$ numbers: V_1, V_2, …, V_m, the volumes of each glass, measured in drops ($1 \le V_i \le 25$, $i = 1, \ldots, m$). The numbering of the glasses follows the rule mentioned above.

Output Specification

Output two numbers in a single line separated by a space. The first number is the minimum time in seconds needed for all the glasses to become full, and the second number is the number of champagne drops that spill on the table.

Sample Input 1

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

Output for Sample Input 1

```
17 0
```

Sample Input 2

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

Output for Sample Input 2

```
27 7
```

D — A Job for a Slicer

You are a slicer (hacker) on board of a rebel cruiser with the mission of freeing an essential ally of the Galactic Empire, but to do that, you have to get access to the main terminal using a special card. The terminal has an internal algorithmic problem and a couple of some input tests, which are daily randomly generated. The terminal expects to get the same output from its internal code and the card's code.

You have just a blank card and the statement for the algorithmic problem that must be solved to print the code on the card. The statement sounds like this:

You are given a rooted tree, where node 1 is the root. The tree has N nodes, and each node has an initial value of 0. You are also given Q queries of 3 types:

- `1 u v`: add value v to all nodes in the subtree with root u;
- `2 x`: print the value of node x;
- `3 r`: make node r the new root of the tree.

Input Specification

The first line contains a single number N, which represents the number of nodes ($2 \leq N \leq 100\,000$).

Each of the next $N - 1$ lines contains two numbers separated by a space: a and b, which means that there is an edge between node a and node b.

The next line contains a number Q, which represents the number of queries ($1 \leq Q \leq 100\,000$).

Each of the next Q lines contains one of the following types of queries mentioned above:

- The first type of query contains three numbers separated by spaces. The first number is 1, which indicates it's the first type of query, the second number is u indicating the root of the subtree, and the third number is the value v that must be added to each node in the subtree ($-1000 \leq v \leq 1000$).
- The second type of query contains two numbers separated by spaces. The first number is 2, which indicates it's the second type of query, and the second number is x indicating the node whose value must be printed.
- The third type of query contains two numbers separated by spaces. The first number is 3, which indicates it's the third type of query, and the second number is r indicating the new root of the tree.

Output Specification

The output should contain a single number for each query of type 2 on a different line, indicating the value of node x.

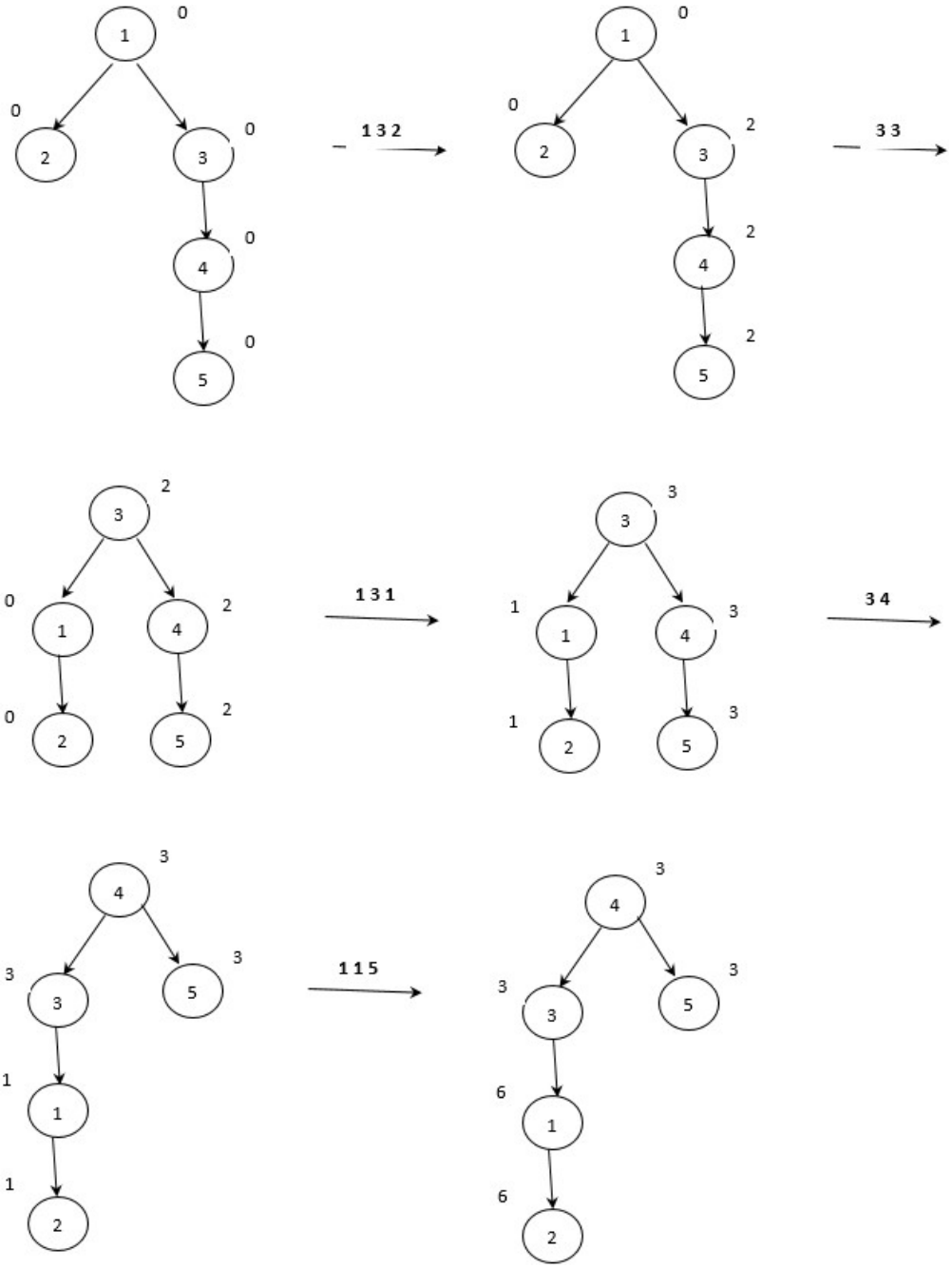
Sample Input

```
5
1 2
1 3
3 4
4 5
10
1 3 2
3 3
1 3 1
3 4
1 1 5
2 1
2 2
2 3
2 4
2 5
```

Output for Sample Input

```
6
6
3
3
3
```

Explanation



As it can be seen here, the answer for the last five queries in order are 6, 6, 3, 3, 3.

E — FIPS Groups

In the research department of an automotive vehicle manufacturer company, a very important task is the testing of compatibility of existing and new electric and electronic components (sensors, controllers, computers, etc.). In order to avoid the manufacturing of a prototype car for every configuration, there is the possibility to examine certain configurations on the existing test cars. These test cars are stored in different locations across Germany, and within one location, they are grouped. These groups are called FIPS Groups. A FIPS Group can contain one or several vehicles.

With the help of a software, a user has the possibility to select from the FIPS Groups any desired group and order them according to their importance. For the ease of understanding, the FIPS Groups are represented by capital letters in this description (but not in the input and output!). For example, if a user selects 5 FIPS Groups [A, B, C, D, E], he will have the possibility to order the groups with the following (sort) operations:

- ONE_LEVEL_UP
- ONE_LEVEL_DOWN
- TO_HIGHEST_LEVEL
- TO_LOWEST_LEVEL

An example explained:

- Input: [A, B, C, D, E], the user would like the [B, D] groups to increase their level of importance with one (ONE_LEVEL_UP) -> Output: [B, A, D, C, E], meaning that group B has taken the place of group A and group D has taken the position of group C.
- Input: [B, A, D, C, E] -> Operation: TO_HIGHEST_LEVEL on [B, A, D] -> Output: [B, A, D, C, E]
- Input: [B, A, D, C, E] -> Operation: TO_LOWEST_LEVEL on [B, A, D, E] -> Output: [C, B, A, D, E]
- Input: [C, B, A, D, E] -> Operation: ONE_LEVEL_DOWN on [C, A, E] -> Output: [B, C, D, A, E]
- Input: [B, C, D, A, E] -> Operation: TO_LOWEST_LEVEL on [C, A, E] -> Output: [B, D, C, A, E]
- Input: [B, D, C, A, E] -> Operation: TO_HIGHEST_LEVEL on [C, A, E] -> Output: [C, A, E, B, D]
- Input: [C, A, E, B, D] -> Operation: ONE_LEVEL_UP on [B] -> Output: [C, A, B, E, D]

Input Specification

The input begins with the comma-separated list of FIPS Groups in their initial succession. Each FIPS Group is represented as a string of at most 10 characters. This is followed by a minus sign ('−') and the sort operations. A sort operation (e.g., `ONE_LEVEL_UP:B,D;`) contains the sort command (ONE_LEVEL_UP in the example), then a colon character (`':'`), then a list of FIPS Groups on which the command should be performed ([B, D] in the example), and a semicolon character (`';'`). The FIPS Groups are separated from each other by comma characters (`','`). There will be at most 1 000 000 sort operations.

Output Specification

Print in one line the order of groups after the required operations are performed as a comma-separated list of strings.

Sample Input

```
A,B,C,D,E-ONE_LEVEL_UP:B,D;TO_LOWEST_LEVEL:B,A,D,E;
```

Output for Sample Input

```
C,B,A,D,E
```

F — Shape ECN

Anna is playing with her new game, “The Word Transformer.” The game has the following objective. A word needs to be transformed into another given word using a minimum number of edits with the following operations: insertion, deletion, or substitution of a single character. Unfortunately, the game had a malfunction, and in order for Anna to win the game, she needs to follow the following rules:

- There are n words to be transformed (with minimal edits).
- The first word needs to be transformed into `ECN`.
- From the second word onward, all the words need to be transformed into a new word that is obtained in the following way:
  - Let d be the minimum number of transformations needed in the previous step.
  - Then, each character of the previously obtained word is replaced with a character that is found at d distance from it in the alphabet. If this character exceeds letter `Z`, it will continue from letter `A`.
  - Example: if the previous minimum number of transformations was $d = 3$, and the previous word was `ABZ`, then the target word is `DEC`.
- The game ends when the minimum number of edits is found for every word.

Input Specification

The first line of the input contains n, the number of words to be transformed ($1 < n \leq 50$). This is followed by n words on separate lines, containing only uppercase letters of the English alphabet. The length of the words is at most 100 characters.

Output Specification

Output one number, the number of appearances of “`ECN`” in the list of target words.

Sample Input 1

```
4
IRIDESCENT
EPIPHANY
SUPINE
LUMINESCENCE
```

Output for Sample Input 1

```
1
```

Sample Input 2

```
11
EPOCH
SONOROUS
EMOTION
HEADTEACHER
RAINBOW
ECN
BEAUTY
BUTTERFLY
LULLABY
SUNSHINE
HOPE
```

Output for Sample Input 2

```
2
```

Explanation of Sample 2

1. `EPOCH ECN` 4 (the minimum number of edits to transform `EPOCH` into `ECN` is 4)
2. `SONOROUS IGR` 8 (the 4th letter after `E`, `C`, and `N` is `I`, `G`, and `R`, respectively)
3. `EMOTION QOZ` 7
4. `HEADTEACHER XVG` 11
5. `RAINBOW IGR` 6
6. `ECN OMX` 3
7. `BEAUTY RPA` 5
8. `BUTTERFLY WUF` 8
9. `LULLABY ECN` 7
10. `SUNSHINE LJU` 7
11. `HOPE SQB` 4

The number of appearances of "`ECN`" is 2.

G — Greek Matchmaker

During the Greco-Persian Wars, many Greek city-states configured their entire society to maximize military proficiency at all costs. Private interests and personal happiness were often subordinated to the good of the public, and the ancient Greek legislators considered that even marriage is a matter of public interest. For example, in Sparta, the laws of Lycurgus of Sparta required that criminal proceedings be taken against those who married too late, as well as against confirmed bachelors. Plutarch, the famous Greek biographer and essayist, reports the peculiar customs associated with marriages and weddings in this ancient period.

In the city of Laodicea ad Sipylum, couples were formed once a year when young men were home from the wars, through an intricate matchmaking dancing ceremony:

1. N girls and N boys, suitable for marriage, were chosen.
2. The high priestess of the Temple of Aphrodite, after praying for visions, would choose a number M, called shuffle rounds.
3. Once M was revealed, the N girls were lined up into a queue according to the alphabetical order of their names. Then the high priestess of the Temple of Aphrodite would arrange the N boys into a parallel queue, in an order she saw fit, in accordance with her visions and divine inspirations.
4. After the queues were formed, the dance ceremony began, everybody dancing in place to the rhythm.
5. Periodically, when he felt inspired, the priest of the Temple of Ares would hit the rhoptron (a special buzzing drum used in Ancient Greece) or blow the salpinx (a trumpet-like instrument of the ancient Greeks). Whenever the rhoptron was hit, the first girl at the top of the girls' queue would go dancing to the back of the queue, and similarly, when the salpinx was blown, the boy from the top of the boys' queue went to the back. There was no specific order or even a fixed distribution on how the sound instruments were played, the priest of the Temple of Ares always acted in the moment, randomly, on how he felt inspired.
6. After the instruments were played M times, i.e., M young people shuffled cyclically to the back of their respective queues, the music was paused for a moment: the girl and the boy on the top of the queues were declared a couple, and they were removed from the queues to go and start preparing for their wedding.
7. Then, the music resumed, and the next couple was determined after M new shuffles.
8. Steps 5–7 were repeated until just one boy and one girl remained in the queues: the final couple.

Let's see an example for $N = 3$ and $M = 3$. Starting from the initial queues from the first two columns, if, for example, first the rhoptron (R) is hit, then the salpinx (S) is sounded twice, the first couple would be (Eileithyia, Philpoemon):

Girls	Boys		Girls	Boys		Girls	Boys		Girls	Boys
Alcippe	Epitrophos	R	Eileithyia	Epitrophos	S	Eileithyia	Haemon	S	Eileithyia	Philpoemon
Eileithyia	Haemon	→	Rhene	Haemon	→	Rhene	Philpoemon	→	Rhene	Epitrophos
Rhene	Philpoemon		Alcippe	Philpoemon		Alcippe	Epitrophos		Alcippe	Haemon

In the second matchmaking round, if the rhoptron is played first, then the salpinx, then the rhoptron again, the second couple would be (Rhene, Haemon):

Girls	Boys		Girls	Boys		Girls	Boys		Girls	Boys
Rhene	Epitrophos	R	Alcippe	Epitrophos	S	Alcippe	Haemon	R	Rhene	Haemon
Alcippe	Haemon	→	Rhene	Haemon	→	Rhene	Epitrophos	→	Alcippe	Epitrophos

The two last persons (Alcippe, Epitrophos) would form the final couple.

One year, while the high priestess of the Temple of Aphrodite was praying for visions and divine inspirations regarding the suitable shuffle round value M and for guidance on how to arrange the boys initially in the queue, Eros, the Greek god of love, visited her. Being after a long night of partying on Mount Olympus, Eros could not be bothered to do the math and reveal the sacred order on how boys should be queued and what is the required divine shuffle number that form the perfect couples. Instead, Eros only provided the high priestess with the perfect couples' list, containing N pairs of names, each pair describing a girl and her match.

It is your task to help the high priestess of the Temple of Aphrodite and do the job of the Olympian Greek Gods! Decide how the boys should be queued and what the smallest positive integer for shuffle rounds is such that the ceremony will lead to the formation of the couples as described by Eros' list.

Input Specification

The first line of the input contains the number N ($2 \leq N \leq 50\,000$), the number of boys, girls, and couples to be. The following N lines describe the perfect couples as intended by the Greek Gods, containing two names per line, that of a girl followed by that of a boy. The names on a line are separated by a space character.

Every girl and every boy has a different name (unique identifier), containing only alphanumeric characters, and every name can be found only once in the input (there is no person belonging to multiple couples).

Output Specification

Output the name of the boys in the order they should be queued, one name per line. In the last line, output the smallest positive integer for shuffle rounds M modulo 1 000 000 007 ($10^9 + 7$) such that the ceremony leads to the formation of the couples as specified in the input.

Sample Input

```
2
Rhene Haemon
Alcippe Philpoemon
```

Output for Sample Input

```
Haemon
Philpoemon
1
```

H — Numbers with Invariant Tail by Squaring

Notice the following interesting relation for these 3-digit numbers:

$$\mathbf{625}^2 = 390\,\mathbf{625} = \mathbf{625} \mod \mathbf{1000}$$
$$\mathbf{376}^2 = 141\,\mathbf{376} = \mathbf{376} \mod \mathbf{1000}$$

Notice also that if an n-digit number a has the property $a^2 = a \mod 10^n$, then also $b = 10^n + 1 - a$ has the same property, $b^2 = b \mod 10^n$. Indeed,

$$\begin{aligned} b^2 \mod 10^n &= (10^n + 1 - a)^2 \mod 10^n \\ &= (1 - a)^2 \mod 10^n \\ &= 1 - 2a + a^2 \mod 10^n \\ &= 1 - 2a + a \mod 10^n \\ &= 1 - a \mod 10^n \\ &= 10^n + 1 - a \mod 10^n \\ &= b \mod 10^n \end{aligned}$$

This explains the connection 625 + 376 = 1001.

Question: Are there numbers having 1000 digits that are equal to their square $\mod 10^{1000}$?

Input Specification

There is no input for this problem.

Output Specification

Print the text "NO" if there are no such numbers. If there exist such numbers, print the first 10 digits of each such number on a separate line.

I — Force Fields

In the future, all cities will be protected by force fields.

We say there is a **safe passage** between two cities if the force fields of the two cities intersect in at least one point.

A group of cities is called a **safe component** if any two cities from the group are connected directly or indirectly by safe passages. A single city is always a safe component if the radius of the force field is not 0.

Given the number n of the cities and their (x, y) coordinates, and assuming each city has a force field of the same radius, answer the following questions:

- Q1: Given the radius of the force fields, how many safe components exist?
- Q2: Given the radius of the force fields and two cities, is there a safe passage between the two cities?

Input Specification

The first line of the input contains an integer n, the number of cities ($1 \le n \le 1000$).

Each of the next n lines contains two integers x_i and y_i, the coordinates of ith city ($0 \le x_i, y_i \le 2000$, $0 \le i \le n - 1$).

The next line contains an integer m, the number of questions ($1 \le m \le 400\,000$).

Each of the next m lines starts with an integer t, the type of question.

- If $t = 0$, there is only a real number r after t, the radius of the force fields.
- If $t = 1$, t is followed by a real number r (the radius of the force fields) and two integers i and j, the indexes of two cities ($0 \le i, j \le$ n $- 1$).

Output Specification

For each of the m questions, output a line containing the answer to the question. For $t = 0$, answer Q1, and for $t = 1$, answer Q2. For Q2, if there is a safe passage between the two cities, output 1, otherwise 0.

Sample Input

```
4
3 3
2 2
3 2
4 2
4
0 0.5
1 10 3 1
1 5.1 2 0
1 0.49 2 3
```

Output for Sample Input

```
4
1
1
0
```

J — Matches

S and P are two strings consisting of lowercase letters of the English alphabet, with lengths N and M ($N \leq 10\,000$, $M \leq 1000$, $N \geq M$).

In string P, we can replace all the letters in a subsequence of choice with the character `'?'`. For example, if P is `'ababu'`, then, by replacing the subsequence `'bab'`, P becomes `'a??u'`. The replaced characters must have consecutive positions. The cost of this operation is the number of modified characters, 3 in the example.

The number of matches of the modified P string in string S is the number of positions on which P matches in S. P matches in S on some position $i \leq N - M$ if any character from P on position $0 \leq j < M$ matches the corresponding letter in S, more precisely, if P_j matches S_{i+j} (using 0-based indexing). A letter matches only itself, while the `'?'` character matches any letter.

What is the minimum cost of an operation made on P such that P has at least K matches in S?

Input Specification

The first two lines of the input will contain strings S and P. The third line contains the natural number K ($K \leq |S| - |P| + 1$).

Output Specification

Print the minimum cost of one operation made on P such that P has at least K appearances in S.

Sample Input 1

```
abecanacc
abac
2
```

Output for Sample Input 1

```
2
```

Sample Input 2

```
ababaiubu
ababu
3
```

Output for Sample Input 2

```
4
```

Sample Input 3

```
xxxzzzyyy
babac
4
```

Output for Sample Input 3

```
5
```

Sample Input 4

```
1. abababa
2. ba
3. 2
```

Output for Sample Input 4

```
1. 0
```

Explanation

In Sample 1, if we replace `'ba'`, *P* becomes `'a??c'` with two appearances in *S*:

```
abecanacc ← S
a??c----- ← first appearance
----a??c- ← second appearance
```

In Sample 2, if we replace `'babu'` in *P*, we get `'a????'` with three appearances in *S*:

```
ababaiubu ← S
a????---- ← first appearance
--a????-- ← second appearance
----a???? ← third appearance
```

There is no solution replacing only three characters.

In Sample 3, replace all characters of *P*, and then there will be 5 appearances in *S*.

In Sample 4, there is no need to change anything, *P* appears three times in *S*.

K — Run-Length Encoding, Run!

Forrest lives in a prehistoric era of "dial-up Internet." Unlike the fast streaming of today's broadband era, dial-up connections are only capable of transmitting small amounts of text data at reasonable speeds. Forrest has noticed that his communications typically include repeated characters and has designed a simple compression scheme based on repeated information. Text data is encoded for transmission, possibly resulting in a much shorter data string, and decoded after transmission to reveal the original data.

Photo by secretlondon123

The compression scheme is rather simple. When encoding a text string, repeated consecutive characters are replaced by a single instance of that character and the number of occurrences of that character (the character's *run length*). Decoding the encoded string results in the original string by repeating each character the number of times encoded by the run length. Forrest calls this encoding scheme *run-length encoding*. (We don't think he was actually the first person to invent it, but we haven't mentioned that to him.)

For example, the string `HHHeelllo` is encoded as `H3e2l3o1`. Decoding `H3e2l3o1` results in the original string. Forrest has hired you to write an implementation for his run-length encoding algorithm.

Input Specification

The input consists of multiple test cases. Each test case is a single line of text. The line starts with a single letter: `E` for encode or `D` for decode. This letter is followed by a single space and then a message. The message consists of 1 to 100 characters.

Each string to encode contains only upper- and lowercase English letters, underscores, periods, and exclamation points. No consecutive sequence of characters exceeds 9 repetitions.

Each string to decode has even length. Its characters alternate between the same characters as strings to encode and a single digit between 1 and 9, indicating the run length for the preceding character.

Output Specification

For each test case, print one line of output. On an input of `E`, output the run-length encoding of the provided message. On an input of `D`, output the original string corresponding to the given run-length encoding.

Sample Input

```
1. E HHHeellloWooorrrrlld!!
2. D H3e2l3o1W1o3r4l2d1!2
```

Output for Sample Input

```
1. H3e2l3o1W1o3r4l2d1!2
2. HHHeellloWooorrrrlld!!
```

L — Dictionary

We wish to create a dictionary of artificial words. The words have to fulfill the following criteria:

- The words contain only letters `a`, `b`, and `c`.
- The length of each word is exactly n.
- A letter never appears on its own; it will always belong to a sequence of 2, 3, or 4 identical letters.
- Any two identical letters from different sequences are separated by at least 4 other letters.

Examples and counterexamples for length $n = 12$:

The following examples of length 12 are words: `aaaabbbbaaaa`, `aaabbbccaaaa`, `aabbccaaaacc`, `aabbccaabbcc`.

The following examples of length 12 are not words:

- `aa`**`a`**`bbb`**`a`**`aaabb`: there are two letters `a` from different sequences that are not separated by at least 4 letters.
- **`aaaaa`**`bbcccaa`: there is a sequence of letters `a` with a length greater than 4.
- `aaaabbcccaa`**`b`**: there is a sequence consisting of a single letter `b`.

The dictionary must contain all words of a given length n.

For a given natural number n, calculate the number of words that will be written in the dictionary.

Input Specification

The input consists of a single line containing a single value of the natural number n ($1 \leq n \leq 2^{63}$).

Output Specification

Output a single line containing the number obtained as the remainder after dividing the number of words of length n by 1 000 000 007.

Sample Input 1

```
6
```

Output for Sample Input 1

```
24
```

Sample Input 2

```
46
```

Output for Sample Input 2

```
317046901
```

Sample Input 3

```
20192019
```

Output for Sample Input 3

```
31255839
```

Explanation

In Sample 1, we have the following 24 words:

`aaaabb`, `aaaacc`, `aaabbb`, `aaaccc`, `aabbbb`, `aabbcc`, `aaccbb`, `aacccc`, `bbaaaa`, `bbaacc`, `bbbaaa`, `bbbbaa`, `bbbbcc`, `bbbccc`, `bbccaa`, `bbcccc`, `ccaaaa`, `ccaabb`, `ccbbaa`, `ccbbbb`, `cccaaa`, `cccbbb`, `ccccaa`, `ccccbb`

In Sample 2, the number of words of length 46 is 1 317 046 908, and the remainder of the division by 1 000 000 007 is 317 046 901.

M — Chess Teams

In a classroom, there are N students. Each of them is either a boy or a girl.

Sometimes, the boys are arguing with each other because of some girl. Other times, some girls fight with each other because of the boys. But a boy never argues with a girl or vice versa. This is why there are some pairs of boys and some pairs of girls who cannot be in the same team. We will call such pairs of students *incompatible*.

You want to select a team for the national chess competition. For this, you will consider each student's ELO rating (a rating that indicates how good a student is at chess). The rating of a team is the sum of the ratings of all its members.

Some subset of students can form a team if

- the number of girls in the team equals the number of boys, and
- there are no two incompatible students in the team.

You must find the maximum rating of a valid team.

Input Specification

The first line of the input contains the number N ($1 \leq N \leq 40$).

Each of the next N lines contains the description of a student: a character representing the gender of the student (`g` for girl and `b` for boy) and an integer E representing their ELO rating ($0 \leq E \leq 5000$).

Then, the next line will contain a number M, the number of pairs of incompatible students ($1 \leq M \leq N \cdot (N-1) / 2$).

Each of the next M lines will contain two integers, denoting the indexes of a pair of students who are incompatible (indexing starts from 1 with the first student). It is guaranteed that for each pair of students, they are either both boys or both girls.

Output Specification

Output a single integer, the maximum ELO rating of a valid team.

Sample Input

```
6
b 253
b 1253
g 5000
g 1
g 2
g 3
3
3 4
3 5
3 6
```

Output for Sample Input

```
6253
```