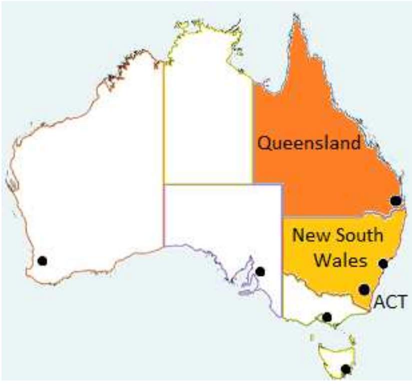


ACM ICPC
South Pacific Region
Divisional Round

August 22, 2015

Central Division Contest Problems

This page has intentionally been left almost blank

A: Selling Numbers

Time Limit: 10 second(s)

Revolutionising telephony is expensive business. That's why young entrepreneur Ace E. Emme is hoping to sell some of his trademarked Global Unique phone numbers first, and then direct the resulting cash at the technical hurdles to see what happens.

Global phone numbers will need to have plenty of digits, which makes it more important than ever to buy a number that is easy to recite from memory. To this end, each phone number is given a Memorisability Score. For a particular phone number, the score is determined as follows:

1. Initialise the score to zero.
2. For each substring of length L, add L to the score if the substring is a palindrome and $L \geq 2$. A palindrome is a sequence that reads the same backwards as forwards.
3. For each pair of non-overlapping substrings A and B, where B appears after A, and each is of length L, with $L \geq 2$, add L to the score for each and every one of the following conditions that holds:
   (a) $A = B$
   (b) $A = B$ and B is adjacent to A
   (c) A is equal to B in reverse.

Mr Emme is interested in pricing the phone numbers, therefore counting how many there are with a particular score is crucial for designating them as Gold Class, Diamond Class and Diamond Class Plus Plus.

Note that each rule on a given substring or pair of substrings is treated independently of the application of this or other rules to other substring(s). For instance, a palindrome of length 5 always contains a palindrome of length 3, as well as a match of rule 3(c). Therefore, the effective score for such a five-character substring will be at least 5 + 3 + 2. This is intentional, as longer patterns appear more lucrative to customers than multiple smaller patterns of the same total length, so a higher score is warranted.

Input

The input contains no more than 11 000 test cases.

Each test case will consist of two integers D $(0 < D < 12)$ and S $(0 \leq S < 1000)$ on a line, separated by a single space. This is a query asking how many phone numbers with D digits are there with Memorisability Score equal to S. Note that phone numbers with leading zeros are considered valid.

The input concludes with a pair of zeros on a line by itself.

Output

For each test case, print a sentence: "`Among D digit phone numbers, there are N with score S.`" Follow the format of the sample output.

Sample Input and Output

Sample Input 1	Output for Sample Input
2 2 3 7 3 0 0 0	Among 2 digit phone numbers, there are 10 with score 2. Among 3 digit phone numbers, there are 10 with score 7. Among 3 digit phone numbers, there are 720 with score 0.

acm International Collegiate Programming Contest 2015 IBM event sponsor
SOUTH PACIFIC REGION

B: Almost an Anagram

Time Limit: 1 second(s)

Andy loves anagrams. For the uninitiated, an anagram is a word formed by rearranging the letters of another word, for example `rasp` can be rearranged to form `spar`. Andy is interested to know if two words are almost anagrams. A word is almost an anagram of another word if:

- one word is shorter than the other by one letter but otherwise contains the same letters in any order; or
- the two words are the same length and their character multisets differ by one character only e.g. "`aaa`" and "`aab`"

Your job is to help Andy to determine if two words are identical, anagrams, almost anagrams or nothing like each other.

Input

The input contains a single test case.

The input will be a single line of text containing a pair of words separated by a single space. The words will be in lower case and will contain alphabetic characters only. Words will contain between 1 and 1000 letters inclusive.

Output

Your program should produce one line of output as follows:

- If the words are identical, output: `word`$_a$ `is identical to word`$_b$
- If the words are anagrams, output: `word`$_a$ `is an anagram of word`$_b$
- If the words are almost anagrams, output: `word`$_a$ `is almost an anagram of word`$_b$
- Otherwise, output: `word`$_a$ `is nothing like word`$_b$

In all cases the first word in the output sentence must be the shorter word or if the words are the same length the first word must be the lexicographically least.

Sample Input and Output

Sample Input 1	Output for Sample Input
`rasp spar`	`rasp is an anagram of spar`

Sample Input 2	Output for Sample Input
`table able`	`able is almost an anagram of table`

Sample Input 3	Output for Sample Input
`sable table`	`sable is almost an anagram of table`

This page has intentionally been left almost blank

acm International Collegiate Programming Contest 2015 IBM event sponsor
SOUTH PACIFIC REGION

C: Who Do You Think You Are?

Time Limit: 2 second(s)

Aunt Clara-May has been taking an interest in the genealogy of the family. She is able to construct a family tree but is getting confused with the relationships between different members of the family.

She wants to identify the following relationships: father, mother, uncle, aunt, son, daughter, nephew, niece, cousin, husband and wife. She also wants to be able to recognise whether the members in the family are related by blood or by marriage (i.e. in-laws), as well as different generations in the family such as grandparents, grandchildren, great grandparents, great grandchildren, great great grandparents, great great grandchildren etc. and different degrees of cousins including levels of removedness (e.g. second cousins-in-law twice removed).

Aunty C-M, as you call her, has some definitions of these relationships but needs your help to write a program to construct the family tree and name the relationships.

You have told Aunty C-M that you will help under the following conditions:

- no second marriages which require step relationships e.g. step-brother and step-father will be recorded
- all children in the family tree will be the offspring of a male father and a female mother who are married
- no marriages between siblings or between cousins of any type have occurred
- all people in the family tree are connected

Further to those conditions, the following definitions apply:

- `father` and `mother` are the parents of a child
- `brother` and `sister` are male and female siblings with the same parents
- `son` and `daughter` are the children of a parent
- `uncle` and `aunt` are the brother and sister of a child's parent
- `nephew` and `niece` are the male and female children of a sibling
- a `grandfather` and `grandmother` are the male and female parents, respectively of a child's parent
- a `great grandfather` and `great grandmother` are the father and mother, respectively, of a child's grandparent
- a `great uncle` or `great aunt` is a sibling to a child's grandparent
- `cousins` (not removed) are at the same level in the family tree
  - first cousins have the same grandparents
  - second cousins have the same great grandparents
  - and so on

- removed cousins are at different levels in the family tree
  - a `first cousin once removed` is the child of one of the first cousins
  - a `first cousin twice removed` is the grandchild of one of the first cousins
  - and so on
- cousin relationships are symmetric e.g. if A is the first cousin twice removed of B, B is also the first cousin twice removed of A
- where a relationship occurs due to marriage of two people the relationship is said to be `in-law`
  - the parent of a person's husband or wife is a `father-in-law` or `mother-in-law`
  - the sibling of a person's husband or wife is a `brother-in-law` or `sister-in-law`
  - the cousin of a person's husband or wife is a `cousin-in-law`

Figure 1 displays the family tree described in the Sample Input. Your program should be able to say that `Claire and Carol are 1st cousins`, `Claire and Diva are 1st cousins 1-time removed`, and `Claire and Chris are cousins-in-law`. Your program should also be able to generate any other relationship combinations when queried.

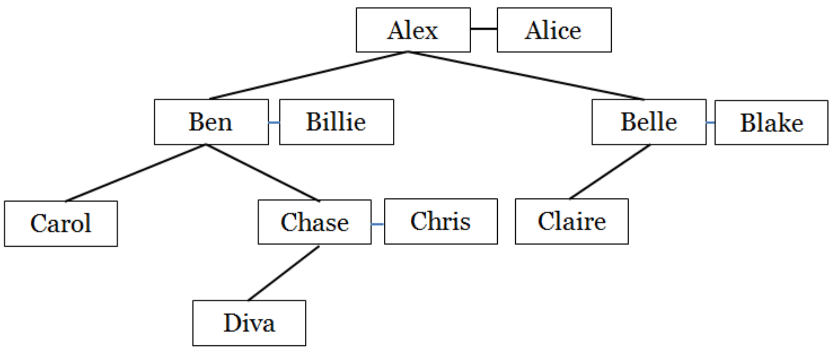


Figure 1: Sample Input

Input

The input contains a single test case.

The input consists of a list of relationships for construction of the family tree. The list of relationships will be followed by a list of queries for which you will name the relationship. Relationships will be provided to infer the gender of all family members.

All relationships will be in lower case and all names will be unique. At most one person in each marriage will have parents present in the input.

The first line of input contains a single integer r ($1 \leq r \leq 200$) being the number of relationships for building the family tree. r lines of relationship definitions follow. Each relationship consists of three alphabetic strings, $name_1$, $name_2$ and *relation*, each separated by a single space. *relation* will be one of `husband`, `wife`, `son` or `daughter`. The relationship line can be read as:

$name_1$ `is the` *relation* `of` $name_2$

The relationships are followed by a line containing a single integer q ($1 \leq q \leq 200$) being the number of queries on the family tree. q query lines follow. Each query line consists of two strings, $name_1$ and $name_2$ separated by a single space. The names in the queries will be contained in the family tree.

Output

For each relationship query, output the relationship between $name_1$ and $name_2$ on a single line.

In the following definitions mandatory items are delimited with `(` and `)`, optional items are delimited with `[` and `]`, options are separated by `|`. Elements which may require repetition (1 to many) are followed by `*`.

- For a spousal relationship i.e. `husband` or `wife`, output a sentence of the following form:

  $name_1$ `is the (husband|wife) of` $name_2$

- For a sibling relationship i.e. `brother` or `sister`, output a sentence of the following form:

  $name_1$ `is the (brother|sister)[-in-law] of` $name_2$

- If the relationship is some kind of cousin, output a sentence which includes the degree of cousinship i.e. `1st`, `2nd`, `3rd` etc. followed by the word `cousins`, then the suffix `-in-law` if and only if the relationship is by marriage and finally the number of times removed (`1-time removed`, `2-times removed`, `3-times removed`, etc.).

  $name_1$ `and` $name_2$ `are (1st|2nd|3rd|...) cousins[-in-law][ (1-time|2-times|3-times|...) removed]`

- If the relationship is aunt, uncle, nephew or niece, the output may require one or more instances of the word `great`.

  $name_1$ `is the [great ]*(aunt|uncle|nephew|niece)[-in-law] of` $name_2$

- Otherwise the relationship will be one of `son`, `daughter`, `father` or `mother`. Relationships which are two generations apart will require the use of the word `grand` before the relationship. Relationships which are more than two generations apart will require the use of one or more instances of the word `great` before the word `grand`.

  $name_1$ `is the [[great ]*grand](son|daughter|father|mother)[-in-law] of` $name_2$

Sample Input and Output

Sample Input	Output for Sample Input
10	Claire and Carol are 1st cousins
Alex Alice husband	Claire and Diva are 1st cousins 1-time removed
Ben Alex son	Claire and Chris are 1st cousins-in-law
Chase Ben son	Billie is the sister-in-law of Belle
Diva Chase daughter	Billie is the mother-in-law of Chris
Carol Ben daughter	Billie is the mother of Chase
Belle Alex daughter	Belle is the aunt of Carol
Chris Chase wife	Blake is the uncle-in-law of Carol
Blake Belle husband	Carol is the niece of Belle
Billie Ben wife	Carol is the niece-in-law of Blake
Claire Belle daughter	
10	
Claire Carol	
Claire Diva	
Claire Chris	
Billie Belle	
Billie Chris	
Billie Chase	
Belle Carol	
Blake Carol	
Carol Belle	
Carol Blake	

This page has intentionally been left almost blank

acm International Collegiate Programming Contest 2015 IBM event sponsor
SOUTH PACIFIC REGION

D: Banking

Time Limit: 1 second(s)

Internet banking sites have a variety of methods to authenticate their users. The methods usually involve passwords or Personal Identification Numbers (PINs) together with a mechanism to verify that a person is attempting to authenticate rather than a computer program.

The Actuarial Commerce Merchant bank has a scheme where, when you login, you are provided with a "pattern word", containing only upper and lower case letters. You must use this pattern word to extract and sum digits from your PIN as follows.

Letters in the pattern word are to be interpreted as numbers, with a (or A) = 1, b (or B) = 2, ... z (or Z) = 26. A lower case letter specifies a count of digits to extract from the PIN while an upper case letter specifies a counts of digits to be skipped. The letters in the pattern word are processed from left to right resulting in a sequence of extracted digits, which are added together to yield a number. You then enter that number into a field on the web page form to authenticate yourself. For example, if your PIN was `1093373`, and the pattern provided to you was `aBcA` you would extract one digit (namely 1) skip two digits (09), extract 3 digits (337) and then skip 1 digit (3), before totalling the extracted digits (1337) and entering 14 into the field on the web page form.

The bank allows you to have a PIN containing up to 256 digits and they intend to provide a pattern word in which the letters, when interpreted as numbers, sum to the length of the PIN. However, sometimes they get this wrong!

Write a program that reads a PIN and a pattern word and outputs the sum of the digits extracted from the PIN if the pattern is valid or outputs `non sequitur` if the length of the PIN and the length indicated by the pattern are different.

Input

The input contains a single test case.

The first line of input will contain an n-digit PIN, $6 \le n \le 256$. The second line will contain an m-digit pattern word containing only upper and lower case letters, $1 \le m \le 256$.

Output

The test case will produce one line of output being either the sum of the extracted digits from the PIN if the pattern word is valid or the text `non sequitur` if the pattern is invalid.

Sample Input and Output

Sample Input 1	Output for Sample Input
092384907653 bGc	23

Sample Input 2	Output for Sample Input
092384907653 bGb	non sequitur

This page has intentionally been left almost blank

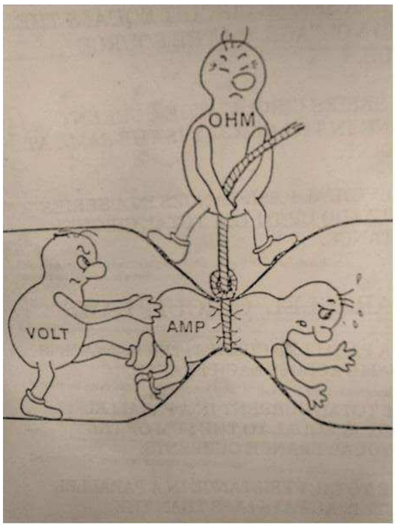
acm International Collegiate Programming Contest 2015 IBM event sponsor

SOUTH PACIFIC REGION

E: Resistance Is (Not) Futile!

Time Limit: 2 second(s)

You have been hired by Acme Circuit Manufacturers to help reduce the number of resistors used in their mass-produced electrical circuits, which will reduce manufacturing costs.



The humble resistor is a small, but crucial component in every electrical circuit. It plays a major role in regulating the flow of electrons (*current*) throughout a circuit by converting electrical energy into kinetic energy (*heat*), and dissipating that heat, in a controlled and quantified manner. We refer to this energy conversion as *resistance*. The unit of measurement to quantify resistance is *ohms*. The higher the ohm value, the higher the resistance.

When deciding on the number and type of resistors to be used, we need to first consider how much current we want within a path of a circuit, and also the potential energy needed to transfer electrons from one point along that path to another (*voltage*). We relate current, voltage and resistance, using a very simple formula known as *Ohm's Law*:

$V = IR$ (where V is voltage, I is current and R is resistance).

For example, let's say that on a given path in our circuit, we have 44558 volts applied and need 10 amperes of current through that path. By applying Ohm's Law (and rearranging the equation to make R the subject), we determine that we need to place a resistor along that path whose resistance is 4455.8 ohms. Problem solved, right?

Wrong.

Unfortunately, ACM only assembles circuits, it does not manufacture the components. This includes resistors. In fact, most electronics companies rely on pre-manufactured resistors. Because of this, there has been a need for standardisation of resistor values. ACM makes use of the E-12 standard range of resistors, so called because there are 12 standard base resistor values that all resistors in that range make use of, namely:

10, 12, 15, 18, 22, 27, 33, 39, 47, 56, 68, 82.

This is referred to as the first *decade* of E-12 resistor values (measured in ohms). The second decade is:

100, 120, 150, 180, 220, 270, 330, 390, 470, 560, 680, 820.

The third and subsequent decades can easily be derived by multiplying each base value by the appropriate power of 10.

The resistances of E-12 resistors are usually only approximately equal to their nominal value but ACM have found a supplier that guarantees exact resistances. ACM wishes to use combinations of these exact resistors to achieve close approximations to actual desired resistances while using the fewest number of resistors. Resistors are always to be connected in series so that the resistance value of a set of resistors is the sum of their resistances. To measure the closeness of an approximation, ACM define the error as the distance of the approximate value (the sum of the resistances) from the target value expressed as a percentage of the approximate value. They wish to ensure that that error is at most 1%.

For example, if we wish to approximate 4455.8 ohms, we could choose the following set of resistors:

3900, 470, 82

as they sum up to 4452 ohms. The error is only $|(4455.8 - 4452)| * 100/4452 = 0.085\%$ which is well within the desired accuracy of 1%. However, a better choice would be:

3900, 560.

While the total resistance of 4460 ohms is not as accurate as that achieved with the previous choice, the error is still well under 1% and, importantly, this combination uses one less resistor (remember, manufacturing costs add up on a large scale).

Your task is to write a program that, given a voltage and current, chooses the best set of resistors to provide the required amount of resistance to within the 1% error as defined above.

Input

The input contains a single test case.

The input has two integer values V $(1 \le V \le 10^9)$ and I $(1 \le I \le 10^7)$. V is the voltage and I is the current.

Output

Output a series of E-12 resistor integer values, from largest value to smallest value, separated by a space, which consists of the lowest number of resistors that approximates the target resistance with an error of at most 1%.

You can use the same resistor value more than once. If you find two or more resistor sets with the same number of resistors that are within the error range, output the set whose sum is closest to the target value. If there are two sets that are the same distance from the target value, output the set whose sum is smaller. If there are still ties, output the set of resistors which is lexicographically least (when the resistors are ordered from largest to smallest).

If there are no possible sets of resistors that allow this resistance, output `Impossible` instead.

Sample Input and Output

Sample Input 1	Output for Sample Input
`50000 5`	`10000`

Sample Input 2	Output for Sample Input
`44558 10`	`3900 560`

Sample Input 3	Output for Sample Input
`1 1`	`Impossible`

F: Protest

Time Limit: 3 second(s)

A group of militant IT students want to march from the Computer Science building to the university Vice Chancellor's office to protest the removal of BASIC as the first-year programming language. The students plan to march arm-in-arm in rows to present a unified front. Once a row locks arms, they will stay that way until their demands are met.

The protest planners have mapped out all the walkways on campus to figure out how many students wide each walkway is. All walkways on the campus allow foot traffic in both directions. The planners can see from the campus map that there is a path from the Vice Chancellor's office from the Computer Science building but they need your help with their plans.

As the protesters' leading algorithmist they need you to find a path from the Computer Science building to the Vice Chancellor's office that maximises the number of students who can walk arm-in-arm for the duration of the walk.

Input

The input contains a single test case.

The first line contains two integers p ($2 \leq p \leq 1\,000$) and w ($1 \leq w \leq 50\,000$) specifying the number of unique walkway end points and the number of walkways on the university campus.

The second line contains two integers c and v ($0 \leq c, v < p; c \neq v$) being the walkway end points for the computer science building and the vice chancellor's office respectively.

The following w lines contain three integers w_{start}, w_{end} ($0 \leq w_{start}, w_{end} < p; w_{start} \neq w_{end}$) and w_{width} ($1 \leq w_{width} \leq 1\,000$) being the start and end points for a walkway and the number of students that will be able to walk arm-in-arm along that particular walkway.

Output

On a single line output the maximum number of students who can walk arm-in-arm from the Computer Science building to the Vice Chancellor's office.

Sample Input and Output

Sample Input	Output for Sample Input
7 11 3 5 0 1 15 0 2 23 1 2 16 1 3 27 2 4 3 2 6 21 3 4 14 3 5 10 4 5 50 4 6 9 5 6 42	16

acm International Collegiate Programming Contest 2015 IBM event sponsor
SOUTH PACIFIC REGION

G: Be Rational

Time Limit: 1 second(s)

Captain Jean-Luc Picard and the crew of the U.S.S. Enterprise NCC-1701-D have been dispatched yet again to negotiate a peace treaty, this time between two warring cultures, the Decimators and the Fractionalists. They have fought for many years over the correct way to represent rational numbers. The Decimators represent each rational number as a possibly repeating decimal number, such `0.444...` (which they write as `0.(4)` with parentheses to denote the repeated part), whereas the Fractionalists represent each rational number as a fraction, such as `4/9`. After a devastating war in which hundreds of millions died, the Fractionalists have won. The crew of the U.S.S. Enterprise has the task of converting all numbers in the treaty to fractional form.

Input

The input contains a single test case.

The input consists of a positive rational number, represented as a possibly repeating decimal number. The whole number part comes first, and is always present. The whole number part may then be followed by both a period and a decimal part. The decimal part may end with a repeating part, which is contained in parentheses. For example, `0.(4)` represents the repeating decimal number 0.444.... There is no whitespace within a line. Each test case is no more than 10 characters long.

Output

Output a single line containing a fraction representing the input rational number. The fraction must be in reduced form i.e. the numerator and denominator contain no common factor.

Sample Input and Output

Sample Input 1	Output for Sample Input
2015	2015/1

Sample Input 2	Output for Sample Input
0.(4)	4/9

Sample Input 3	Output for Sample Input
3.(142857)	22/7

Sample Input 4	Output for Sample Input
9.(9)	10/1

This page has intentionally been left almost blank

acm International Collegiate Programming Contest 2015 IBM event sponsor
SOUTH PACIFIC REGION

H: Shelob's Lair

Time Limit: 10 second(s)

Sam Gamgee and Frodo Baggins are trapped in Shelob's lair. Shelob is a gigantic spider who lives in the caves at the edge of Mordor. Sam and Frodo are Hobbits, which means that they are little people with hairy feet.

The cave is a large rectangular cavern and Shelob has cast many great webs in the cave, and now Frodo and Sam (who are at the South wall of the cave) must reach the North wall to escape. If Sam and Frodo touch any of the web, they will become stuck and Shelob will come and eat them.

Sam has a magic sword, Sting, that can cut through Shelob's web. However, Frodo is poisoned and Sam is exhausted from their adventures, so Sam only has the strength to make one vertical slice through the web once. A well-chosen slice at a point will cut through all of the webs that pass through the point, allowing Sam and Frodo to pass. Sam and Frodo, being little people, can fit through an infinitely small slit.

Given the locations of all the webs in the cave, determine if it is possible for Sam and Frodo to escape.

We suppose that each web is a vertical sheet that runs from one point (given by Cartesian coordinates) to another point. The webs are fixed at the roof and the floor of the cave, and run in a straight line between the two points. Multiple webs can cross one another (Shelob is a skilled web spinner) and if Sam were to slice exactly where they cross he could slice all of the webs at once. No web touches the North or South wall of the cave. Sam is also able to cut at precisely the point one or more webs connect to the East or West walls of the cave. You may treat Frodo and Sam as a point, so they can fit through the vertical cut and can fit between two webs that do not intersect.

Input

The input contains a single test case.

The first line consists of three integers, w (the width of the cave), d (the depth of the cave), and n (the number of webs that are cast), where $1 \leq w, d \leq 1000$ and $1 \leq n \leq 500$.

Next, n lines follow where each line contains 4 integers, x_1, y_1, x_2, y_2, where $0 \leq x_1, x_2 \leq w$ and $0 < y_1, y_2 < d$. (x_1, y_1) is the Cartesian coordinates of one end of the web, and (x_2, y_2) is the Cartesian coordinate at the other end of the web. The coordinates are arranged so that the South-West corner of the cave is the point $(0, 0)$, and the North-East corner is the point (w, d).

Output

If it is possible for Frodo and Sam to reach the North wall, making at most one slice through the web, print the line:
`We can make it Mr Frodo!`

If it is impossible for Frodo and Sam to reach the North wall without making more than one slice through the web, print the line:
`We're doomed Mr Frodo!`

Sample Input and Output

Sample Input 1	Output for Sample Input

`5 5 3`
`0 3 5 3`
`0 1 5 4`
`0 2 5 4`
	`We're doomed Mr Frodo!`

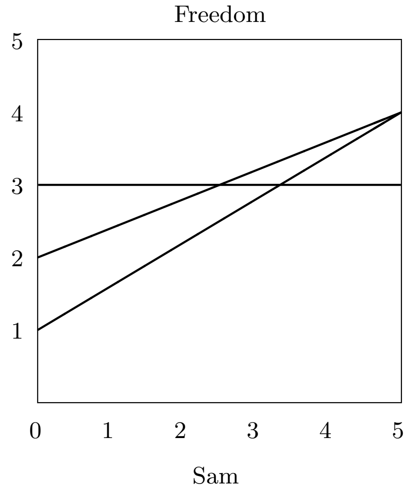
Sample Input 2	Output for Sample Input

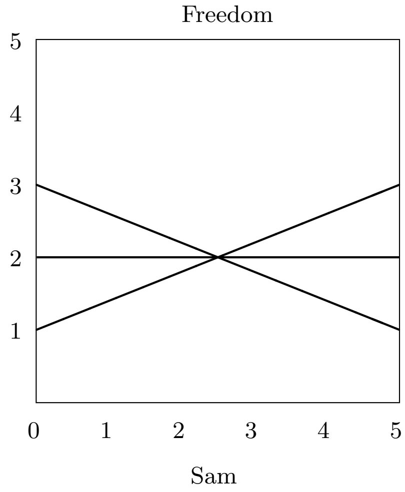
`5 5 3`
`0 1 5 3`
`0 3 5 1`
`0 2 5 2`
	`We can make it Mr Frodo!`

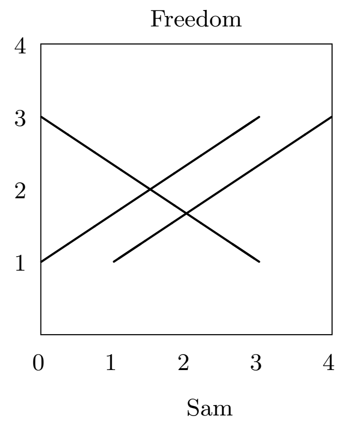
Sample Input 3	Output for Sample Input

`4 4 3`
`0 1 3 3`
`0 3 1 3`
`4 3 1 1`
	`We can make it Mr Frodo!`

The following diagrams are images for the Sample Inputs:







I: Diana and the Golden Apples

Time Limit: 20 second(s)

The famously fleet-of-foot Roman huntress Diana has agreed to marry any man who can beat her or even equal her in a running race. A challenger, Prince Humperdonkey of Troy, is intending to beat her in a race by leaving golden apples along the track. He believes she will be tempted to pick them up, thereby slowing her down enough that he will be able to beat her. Little does he know that Diana, who has no wish to marry anyone at present (and certainly not the loathsome Humperdonkey) is an ICPC competitor who is perfectly able to compute exactly how many golden apples she can pick up while still winning the race. You are Diana and your job is to get as rich as possible while remaining single.

Input

The input contains a single test case.

The first line of input contains 5 space-separated integers: $1 \leq L \leq 1\,000$, the length of the race in units of 100 m; $10 \leq T_d, T_h \leq 30$ the time in seconds that it takes Diana and Humperdonkey respectively to run 100 m; $0 \leq N \leq 1000$ the number of golden apples Humperdonkey has placed on the race track; and $0 < d \leq 10$, the extra time in seconds that Diana takes to cover 100 m for each additional kilogram of gold that she is carrying.

This is followed by N lines, each with 2 space-separated integers $0 < w_i \leq 50$, the weight of the i^{th} apple in kg and $0 \leq x_i < L$, the distance of the i^{th} apple from the start of the track in units of 100 m.

Output

A single integer $W \geq 0$, being the maximum weight in gold apples that Diana can be carrying when she crosses the finish line if she is to finish ahead of Prince Humperdonkey. If Diana is unable to beat Humperdonkey the output line should instead be

```
Diana marries Humperdonkey
```

Sample Input and Output

Sample Input 1	Output for Sample Input
20 10 16 4 2 2 8 3 9 4 10 30 18	5

Sample Input 2	Output for Sample Input
16 18 18 0 2	Diana marries Humperdonkey

This page has intentionally been left almost blank

SOUTH PACIFIC REGION

J: Painting Floors

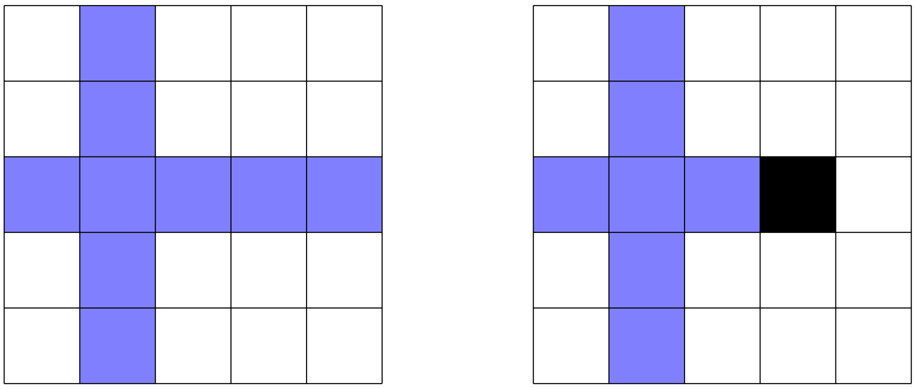
Time Limit: 1 second(s)

I really need to paint my floor! My floor is rectangular and has some furniture on it. Instead of hiring someone to paint it for me, I decided to do it myself. So I went out the local paint store, Antonio's Colourful Masterpieces, and asked for some paint. The man behind the counter asked me if I would like to try some experimental paint. Curious as to what could be experimental about paint, I said, "Yes!" I then purchased 1 000 paint cans and went home (it was on a very good sale).

When I got home, I opened the instruction manual for the paint and was extremely surprised:

> "When you pour the whole can of paint onto the ground, it will fill in the 1×1 block of floor it is in and then will expand out and fill every square that is in the same row or same column as the original 1×1 block so long as there is not an obstacle in the way."

For example, in the left room, the paint is poured in the third row and second column and it fills in coloured squares. In the right room, the paint is poured in the same square, but it only goes until it hits the black obstacle (piece of furniture).



You may not pour paint onto any of the furniture (obviously!). The goal is to paint every square in the room that is not furniture. Painted squares do not become obstacles for future pours of paint and it is okay to paint squares multiple times.

For small rooms, I can easily figure out how to paint the rooms with this experimental paint, but for large rooms, I'm worried that I will run out of paint before I finish the floor! Can you tell me where to pour the paint? You do not need to give me an optimal solution, but you must give me a solution that uses no more than the 1 000 paint cans that I purchased. It is guaranteed that 1 000 paint cans are enough to paint the floor.

Input

The input contains one test case.

The first line will contain two integers m and n ($1 \le m, n \le 500$) being the dimensions of my room in metres. The next m lines will contain n characters each. These lines will show the layout of my room. The map of the rooms will only contain the characters '`.`' and '`x`'. An '`x`' denotes an obstacle in the room and a '`.`' denotes no obstacle. There will be at most 500 `x`'s in the input.

Output

The output will consist of several lines. In each line, output two integers: the row and the column of where to pour the ith can of paint. The rows and columns are 1-based from the top-left corner. The number of lines of output must be no more than 1 000. The lines need not be in any particular order. Any valid output will be considered correct.

Sample Input and Output

Sample Input 1	Output for Sample Input
5 5	1 1 2 2 3 3 4 4 5 5

Sample Input 2	Output for Sample Input
7 7 .... .xxxxx. .xx.xx. .x...x. .xx.xx. .xxxxx.	1 1 4 4 7 7

K: Folding Code

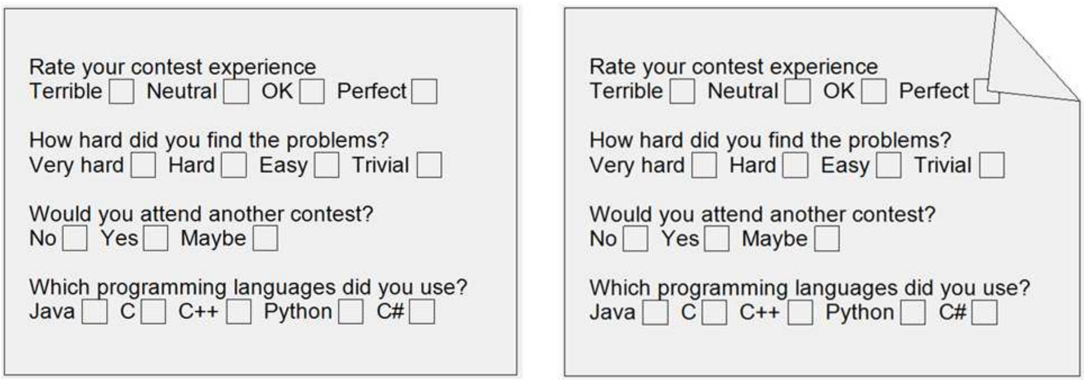
Time Limit: 1 second(s)

Members of the Paperless University programming contest team were used to doing all their work digitally. Imagine their surprise at the end of a programming contest when the organisers asked them to fill in an evaluation form on paper!!! What were they to do? Of course they didn't carry pens or pencils, and circumstances were such that borrowing was not an option. They did wish to submit evaluations. Fortunately, the evaluation form was multi-choice, so all they needed to do was to indicate their preferred option for each question. The team worked out a way of 'filling' in the form.

To answer a question they folded over one of the corners of the page to put it in the preferred answer square and creased the paper along the (straight) fold line. Choice of corner was arbitrary. After answering a question the paper was flattened again. The process was repeated for each answer they wished to 'tick'. The result was a sheet of paper with one fold line for each answer the coder chose to provide.

Reading back the code was easy - make each fold and see where it pointed. Except ... the organisers of the programming contest were digital enthusiasts too. Their process was to scan and destroy all evaluation forms, then analyse the scans. Luckily the scanned forms showed shadows on the fold lines. The organisers wrote a program that allowed an operator to note the points at which fold lines intersected the edge of the paper.

Your task, given a list of fold line intersection points and information about the locations and sizes of question answer options, is to complete the decoding of the evaluation forms. Here is an example evaluation form. In the right image the top right corner has been folded over to provide the answer "Perfect" to the first question.



Input

The input contains a single test case.

The first line of input has four integers: W, H $(100 \leq W, H \leq 1\ 000)$, Q $(0 < Q \leq 10)$ and F $(0 < F \leq 100)$: the width and height of the form in mm; the number of questions on the form; and the number of folds to decode, respectively.

This is followed by Q question descriptions. The first line of each question description has 5 integers, A, x, y, w and h, followed by the text of the question. A is is the number of answers to the question $(0 < A \leq 10)$. The four values x, y, w and h are the coordinates of the top left corner and the width and height of an enclosing rectangle for the question text $(0 < x + w < W)$ and $(0 < y + h < H)$.

The next A lines hold answer descriptions. Each has 8 integers x_1, y_1, w_1 and h_1 defining an enclosing rectangle for the answer text and x_2, y_2, w_2 and h_2 defining the rectangle that is the 'tick' box; followed by the text of the answer. (x_1, y_1) and (x_2, y_2) are the top-left corner of the respective rectangles and w_1, w_2, h_1 and h_2 are the widths and heights of the two rectangles. All rectangles will have positive area and will fit within the dimensions of the page. No rectangles can overlap each other but they may touch. All question and answer strings are non-empty.

After the text descriptions are F lines, each describing one fold in the form. The lines each hold 4 integers: x_1, y_1, x_2, y_2 being the $x-$ and $y-$coordinates of the points at which a fold meets an edge of the paper. All input items are single space separated on their lines.

Fold lines are always between adjacent edges – they will not pass through a corner – and the corner between is the pointer. Each box will be ticked by at most one fold. The number of folds, F, will be less than or equal to the total number of answer boxes.

The coordinate system for the form is in millimetres; (0, 0) is the top left corner; (w, h) is the bottom right corner. Intersection points with an edge will have one coordinate that is exactly 0 (top or left), w (right) or h (bottom). The length of text in a question or answer is never more than 100 characters. You may assume all folds to be valid i.e.: they point into a tick box - and point cleanly inside the box (by at least 0.1mm).

Output

The output for each form should consist of one line per question. For each question the line should be the question text followed by a colon, a space and then a list of semicolon and space separated answers to that question. The answers should be listed in the order given in input. For some questions there may be no answer. For some questions there may be more than one answer.

Sample Input and Output

Sample Input

```
210 150 3 3
4 9 19 121 10 Rate your contest experience
9 29 33 10 43 29 10 10 Terrible
57 29 32 10 90 29 10 10 Neutral
104 29 16 10 121 29 10 10 OK
135 29 31 10 167 29 10 10 Perfect
4 9 49 150 10 How hard did you find the problems?
9 59 42 10 52 59 10 10 Very hard
66 59 22 10 89 59 10 10 Hard
103 59 23 10 127 59 10 10 Easy
141 59 27 10 169 59 10 10 Trivial
5 9 109 182 10 Which programming languages did you use?
9 119 22 10 32 119 10 10 Java
46 119 9 10 56 119 10 10 C
70 119 20 10 91 119 10 10 C++
105 119 31 10 137 119 10 10 Python
151 119 14 10 166 119 10 10 C#
0 50 35 0
210 105 180 150
0 113 27 150
```

Output for Sample Input

```
Rate your contest experience: Terrible
How hard did you find the problems?:
Which programming languages did you use?: Java; C#
```