



Maratona Inter-Universitária de Programação

20TH OF OCTOBER - 10:00 TO 15:00



DEPARTAMENTO DE CIÊNCIA DE COMPUTADORES
FACULDADE DE CIÊNCIAS DA UNIVERSIDADE DO PORTO

Contents

	Page
Information	
Scientific Committee	2
Local Organization Committee	2
Compilers	3
Compilation Constraints	3
Runtime constraints	3
About the input/output	3
Documentation	3
Problems	
Problem A: Braincrash	4
Problem B: The Right Job	6
Problem C: Humanitarian Logistics	10
Problem D: Ferry Boats	12
Problem E: Quick Answers	14
Problem F: Tiled Wall	16
Problem G: The Open Day	18
Problem H: Caesar's Orders	20
Problem I: Flying Balloons	22

Scientific Committee

- Alexandre Francisco (Instituto Superior Técnico)
- Ana Paula Tomás (Universidade do Porto)
- André Restivo (Universidade do Porto)
- Cristina Vieira (Universidade do Algarve)
- Fábio Marques (Universidade de Aveiro)
- Hugo Vieira (Universidade de Lisboa)
- Luís Paquete (Universidade de Coimbra)
- Margarida Mamede (Universidade Nova de Lisboa)
- Paul Crocker (Universidade da Beira Interior)
- Pedro Guerreiro (Universidade do Algarve)
- Pedro Ribeiro (Universidade do Porto)
- Rui Mendes (Universidade do Minho)
- Simão Sousa (Universidade da Beira Interior)

Local Organization Committee

- Ana Paula Tomás
- Fernando Silva
- José Paulo Leal
- Pedro Ribeiro
- Hugo Ribeiro
- Alexandra Ferreira
- Joana Dumas

Compilers

language	compiler version	compile cmd	execute cmd
C:	gcc 4.7.2	gcc <i>name.c</i> -lm	<i>a.out</i>
C++:	g++ 4.7.2	g++ <i>name.cpp</i> -lm	<i>a.out</i>
Java:	javac 1.7.0_07	javac <i>name.java</i>	<i>java -Xmx256M -Xss256M name</i>

Compilation Constraints

- **Maximum compilation time:** 60 seconds
- **Maximum source code size:** 100 KB
- Every source code must be submitted in a single file
- In case of Java submissions, the `.java` file has to have the same name as the class that contains the main method. There is no limit for the number of classes to be contained in that file.

Runtime constraints

These limits apply to all problems.

- **Maximum CPU time:** 2 seconds
- **Maximum Memory:** 256MB

About the input/output

- All lines (both in the input and output) should be ended by the newline character (`'\n'`)
- Except when explicitly stated, single spaces is used as a separator.
- No line starts or ends with any kind of whitespace.

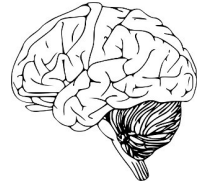
Documentation

Language documentation is available, namely:

- man pages for C/C++ function (*using the command line*)
- C++ STL documentation (*bookmarked on your browser*)
- Java SE 7 Documentation (*bookmarked on your browser*)

Problem A: Braincrash

If you came to this contest, it's because you like programming. And, if you like programming, you must love programming languages. You probably agree with me that it's a pity that we don't have time to learn, to experiment and to play with all those fantastic languages around us. We know: *"so many languages, so little time"*.



Recently, I came across Braincrash (BC), reputedly the simplest language of them all: it runs on a vocabulary of only two symbols: the exclamation mark ('!') and the underscore ('_') and it is Turing complete. The language has no syntax, meaning that any string formed by only underscores and exclamation marks is a valid BC program, syntactically speaking. The semantics is defined in terms of another beautiful language, Brainfuck (BF). BF uses eight symbols, `+-><.,[]`, (plus sign, minus sign, greater than, less than, dot, comma, open square bracket, close square bracket), which makes it a language three orders of magnitude more complicated than BC.

The procedure for translating from BC to BF is rather straightforward: just align the BC program with an infinite repetition of the string `"+-><.,[]"`, and pick those characters that are aligned with the exclamation mark. Here's an example:

```
-----!!-----!!_!  
+-><.,[]+-><.,[]+-><.,[]+-><.,[]
```

This gives the following BF program:

```
,[.,]
```

which any BF fan easily recognizes as the cat program, that copies from the standard input to the standard output.

Task

Your task is to write a program that translates a BF program into the shortest equivalent BC program. In the BF program, any char which is not in the set `+-><.,[]` is considered a comment and should be ignored by the translation.

Input

The input is a BF program, with an undetermined number of lines. The BF program is not empty, even after removing all the comment characters.

Output

The output file contains a single line with the BC program.

Constraints

The maximum size of the input (including newline characters) is 13,000 characters.

Input example

```
,[.,]
```

Output example

```
-----!!-----!!_!
```

(this page is intentionally left blank)

Problem B: The Right Job

In the likely event that your team will rank among the first at MIUP 2012, very soon you will be contacted by the big international software companies, all eager to draw on your programming talent. Some will offer a bigger salary, others a fantastic working environment, a challenging project, the opportunity to work with a new technology, or the prospect of a brilliant international career. You will be at a loss, not knowing which offer to accept, fearing to take the wrong decision.

Fear no more! Being a programming person, just write a program to decide for you. May we suggest the Analytical Hierarchy Process, AHP for short? We'll explain it with an example, which actually can apply to you.

Suppose that after some soul-searching, you have found that you pursue four objectives for your first job as a software developer: a large salary (obj_1), a challenging project (obj_2), a high quality of living in the city to where you will be relocated (obj_3), and a lot of traveling (obj_4).

Not all these objectives are worth the same, for you. For example, in your opinion, a high salary is twice as important as a challenging project and four times as important as a lot of traveling.

You can write down a matrix a , where a_{ij} means that for you, objective i is a_{ij} times more important than objective j . In this matrix, $a_{ii} = 1$ and $a_{ij} \times a_{ji} = 1$, of course.

Objective	1	2	3	4
1	1	2	3	4
2	1/2	1	5	7
3	1/3	1/5	1	3
4	1/4	1/7	1/3	1

Next, you normalize the matrix, by dividing each value by the sum of its column. Then you compute the weight of each objective as the sum of each line divided by the number of columns. Easy?

Objective	1	2	3	4	weight
1	0.4800	0.5983	0.3214	0.2667	0.4166
2	0.2400	0.2991	0.5357	0.4667	0.3854
3	0.1600	0.0598	0.1071	0.2000	0.1317
4	0.1200	0.0427	0.0357	0.0667	0.0663
Sum	1.0000	1.0000	1.0000	1.0000	1.0000

Now suppose you got offers from IBM (job_1), in Zurich, from Microsoft (job_2), in London, and from Google (job_3) in Mountain View. In your perspective, job_1 is twice as important in terms of salary as job_3 and four times as important as job_2 , and job_3 is six times as important in terms of salary as job_2 :

Salary(obj_1)	job_1	job_2	job_3
job_1	1	4	2
job_2	1/4	1	1/6
job_3	1/2	6	1

Repeating the same steps for the other 3 objectives, normalizing the 4 matrices and calculating the weights, we get the preferences of the jobs in respect to the objectives:

	obj_1	obj_2	obj_3	obj_4
job_1	0.5222	0.2364	0.1524	0.2014
job_2	0.0955	0.0623	0.7208	0.6806
job_3	0.3823	0.7013	0.1268	0.1179



The estimation of the preferences of each job in respect to each objective (the weight) is given by the average of the $a_{i,j}, j = 1, \dots, 3$ of the normalized matrix.

Finally, to compute the combined scores of each job, you must add the product of the objective weight and preference. For example, the score of job_1 is given by:

$$score(job_1) = 0.5222 \times 0.4166 + 0.2364 \times 0.3854 + 0.1524 \times 0.1317 + 0.2014 \times 0.0663.$$

Task

Your task is to write a program that given the objectives and their relative worth and the job offers, and you appreciation of their relative value for each objective, ranks the job offers using the AHP method.

Input

The first line contains two integers, **N** and **M**, separated by a single space, indicating respectively the number of objectives and number of jobs.

The second line contains a positive integer **P** that indicates how many pairwise objectives comparison will follow, followed by **P** lines with 3 integers in the form “**A B I**”, indicating that objective *A* is more important *I* times than objective *B*. The value of the missing pair comparisons is 1. The pairs can come in any order and the same pair of objectives cannot appear multiple times.

Then come exactly **N** groups of pairwise jobs comparisons, one for each of the objectives. The *i*-th group corresponds to objective *i*. The input format is the same as the pairwise objectives comparison.

Output

The output file consists of a single line (including the newline character) displaying the number of the *M* jobs ordered by descending score, and separated by single spaces. You can be assured that there will never be ties in the combined scores of two different jobs.

Constraints

$2 \leq \mathbf{N}, \mathbf{M} \leq 50$ number of objectives and number of jobs
 $2 \leq \mathbf{I} \leq 9$ intensity of importance

(the input and output examples are on the next page)

Input example

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

Output example

```
3 1 2
```

(this page is intentionally left blank)

Problem C: Humanitarian Logistics

A humanitarian association received a shipment of containers with goods to be distributed among the population of a remote village, victim of cataclysm. All containers have been unloaded in the same day on a distant seaport from the village and the only available transport is an old truck, that takes one day for each round trip. This means that the truck can deliver a single container each day. Since each container has perishable goods, with a common expiration date, the association must devise a strategy to transport and distribute these goods, minimizing the loss.



Task

Given N containers with expiration dates $\mathbf{T}_1, \dots, \mathbf{T}_N$ and values $\mathbf{V}_1, \dots, \mathbf{V}_N$, your task is to find the optimal scheduling for the transportation such that the loss is minimum, i.e., the sum of values \mathbf{V}_i for all containers not delivered in time is minimum. Assume that the expiration date is in days counted from the unloading at the seaport, i.e., a container i with $\mathbf{T}_i = 2$ means that it must be delivered at most on the second day after being unloaded at the seaport. When deciding among containers with the same value, you should consider first the one with smaller id i .

Input

The number of containers N in the first line followed by N lines with the information for each container. The information for each container consists of the expiration date \mathbf{T}_i and the value \mathbf{V}_i , separated by a single space.

The containers come ordered by their ids. That is, the first container has $id = 1$, the second container has $id = 2$ and so on until the last line of input which describes container with $id = N$. There can be several containers with the same expiration date, and several containers with the same value.

Output

The list of containers, one per line, to be delivered in time, sorted accordingly to their id.

Constraints

- | | |
|------------------------------------|-----------------------------------|
| $1 \leq N \leq 100,000$ | Number of containers |
| $1 \leq \mathbf{T}_i \leq N$ | Expiration date of each container |
| $1 \leq \mathbf{V}_i \leq 100,000$ | Value of each container |

(the input and output examples are on the next page)

Input example 1

7
3 60
3 40
3 80
5 70
5 85
5 90
7 10

Output example 1

1
3
4
5
6
7

Input example 2

7
1 10
4 10
2 5
1 20
1 30
5 30
3 20

Output example 2

2
3
5
6
7

Problem D: Ferry Boats

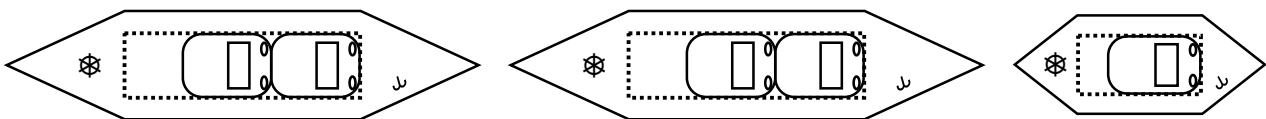
In the summer lots of people go on vacation to the Mediterranean islands. Ferry boats are one of the preferred means of transportation used to get there, since they allow people to take their cars with them. Ferry boat companies manage a huge number of reservations during the summer and one of the main business concerns is how to optimize ferry cargo.



Boats used by ferry companies can carry lots of weight, practically unlimited, but parking space is strictly limited. In general ferry companies use different kinds of boats to serve as the means of transportation, each one differing in the amount of centimeters of parking space. To help the ferry management process, each ferry reservation includes the length of the vehicle that will be transported. Reservations are handled on a first come, first served basis. In order to optimize cargo, ferry companies decide upon which type of boats will be used depending on the amount of parking space wasted in transporting the cars, i.e., the amount of unused parking space, regardless of how many boats have to be used throughout the summer. Cars are parked so close to each other that no space is left between them. Ideally the parking space is completely used up, which means there is no space waste at all.

You have been hired by a ferry boat company that is interested in finding out what is the minimum amount of parking space wasted in all ferry transports that depart from the same origin port and that are directed to the same destination port, during all summer, given the parking capacity of all boats that can be used and the order and length of the vehicles that will be transported. This company in particular operates with an unlimited fleet of boats since they reuse their own repeatedly and rent boats of other companies whenever necessary. There is always at least one type of boat that can carry any kind of car.

For the sake of illustration consider that you are given two types of boats, one with capacity 400 and the other with capacity 800, and a sequence of 5 cars all with length 300. The minimum waste to transport this sequence is 500, obtained, for example, transporting the first car in a boat of capacity 400 (waste 100), transporting the second and third cars in a boat of capacity 800 (waste 200), and transporting the fourth and fifth cars in a boat of capacity 800 (waste 200) as illustrated in the figure below.



Task

Write a program that, given the distinguished set (all distinct values) of capacities of the types of boats that can be used (each type having an unlimited available fleet), and the sequence of lengths of the cars that will be transported, computes the minimum amount of parking space wasted in transporting the cars on the boats, without ever exceeding the parking capacity. The amount of parking space wasted is given by the sum of all unused centimeters in all boat travels necessary to transport all the cars.

Input

The first line contains an integer value that specifies the number of boats \mathbf{B} followed by an integer value that specifies the number of cars \mathbf{C} , separated by a single space. The following \mathbf{B} lines contain \mathbf{B} distinct integer values, each one corresponding to a boat parking space capacity (in centimeters): $\mathbf{S}_1, \dots, \mathbf{S}_\mathbf{B}$. Then, the following \mathbf{C} lines contain \mathbf{C} integer values, each one specifying the length of a car (in centimeters): $\mathbf{L}_1, \dots, \mathbf{L}_\mathbf{C}$. The order in which the cars are served is given by the ordering of the lengths in the input. At least one of the boat capacities is greater than or equal to any car length: $\max(\mathbf{L}_1, \dots, \mathbf{L}_\mathbf{C}) \leq \max(\mathbf{S}_1, \dots, \mathbf{S}_\mathbf{B})$.

Output

The output consists in a single line containing an integer value \mathbf{W} that identifies the minimum amount of parking space wasted (in centimeters).

Constraints

$1 \leq \mathbf{B} \leq 100$	Number of ferry boats
$1 \leq \mathbf{C} \leq 100,000$	Number of cars
$50 \leq \mathbf{S}_{1 \leq i \leq \mathbf{B}} \leq 1,000$	Boat capacities
$50 \leq \mathbf{L}_{1 \leq i \leq \mathbf{C}} \leq 1,000$	Car lengths
$0 \leq \mathbf{W} < 50,000,000$	Parking space waste

Input example 1

2 5
400
800
300
300
300
300
300
300

Output example 1

500

Input example 2

3 3
400
800
600
300
400
200

Output example 2

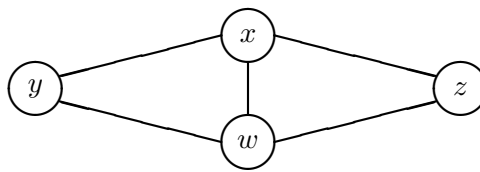
100

Problem E: Quick Answers

The Republic of Earthquakes is a country on an island in the Pacific Ocean. Unfortunately, earthquakes occur regularly and some are very destructive. So, there is a department specially prepared to assist the population, where decisions such as which resources are moved from a location to another are taken with utmost speed. The problem is that, immediately after an earthquake, the information about the damages is vague. For instance, they usually know how many roads are no longer passable much before they can identify them. Nevertheless, quick answers are needed.



All roads have two directions. Besides, before an earthquake, there is at least one path between any two locations. Given the number of roads that have been destroyed (in both directions) and two distinct locations, your task is to determine whether or not there must still be a path between the locations.



Let us see an example with the four locations and the five roads depicted in the figure above. When an earthquake destroys two roads:

- Whichever the destroyed roads are, there has to be a path between x and w .
- There may or may not be a path between x and y . If, for instance, the destroyed roads are (x, y) and (x, z) , the path $x w y$ exists. But there is no path between x and y when roads (x, y) and (y, w) are the not passable ones.

Task

Given the set of locations, the set of roads, the number of roads destroyed by the earthquake, and two distinct locations, the goal is to find out whether or not there must be a path between the latter.

Input

The first line of the input has two integers: \mathbf{L} and \mathbf{R} . \mathbf{L} is the number of locations and \mathbf{R} is the number of roads. Locations are identified by integers, ranging from 0 to $\mathbf{L} - 1$. Each of the following \mathbf{R} lines contains two distinct integers, l_1 and l_2 , which indicate that there is a road between locations l_1 and l_2 .

The last line has three integers, \mathbf{D} , l' and l'' , which represent the question: When an earthquake destroys \mathbf{D} roads, must there still be a path between locations l' and l'' ?

Integers in the same line are separated by a single space.

Output

The output has a single line with the word **Yes**, if there must still be a path between the locations, or **No**, otherwise.

Constraints

- $2 \leq \mathbf{L} \leq 2,500$

Number of locations
- $1 \leq \mathbf{R} \leq 10,000$

Number of roads
- $1 \leq \mathbf{D} \leq \mathbf{R}$

Number of destroyed roads

Input example 1

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

Output example 1

Yes

Input example 2

7 9
0 2
0 3
5 0
4 1
3 5
3 4
5 2
6 4
1 6
1 6 2

Output example 2

No

Problem F: Tiled Wall

Jake was having a nice dinner with his girlfriend in one of their favorite restaurants. She was talking about feelings or something, and as it was often the case, Jake's mind started wandering.

The wall behind her caught his attention. It was one of those old looking walls composed of tiles with different sizes and shapes. The wall was huge and he started thinking about the ammount of work that was put into building such a structure.

As he slipped deeper and deeper into his own imagination, he suddenly realized that the wall was a sham. He could see it clearly now. The wall was made out of much larger rectangular blocks, all with the same size and orientation, and each one of them was composed of several smaller tiles. The magic was gone!

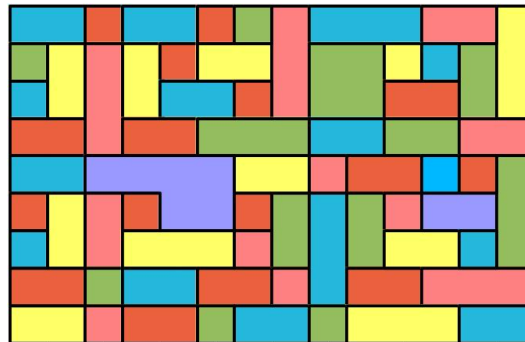


Figure 1: A large wall composed of small tiles

"Were you listening to anything I said?", she asked. He nodded, as he tried to get the waiter's attention, "Check please!". There was only one thing on his mind now: How many blocks was the wall really made of?

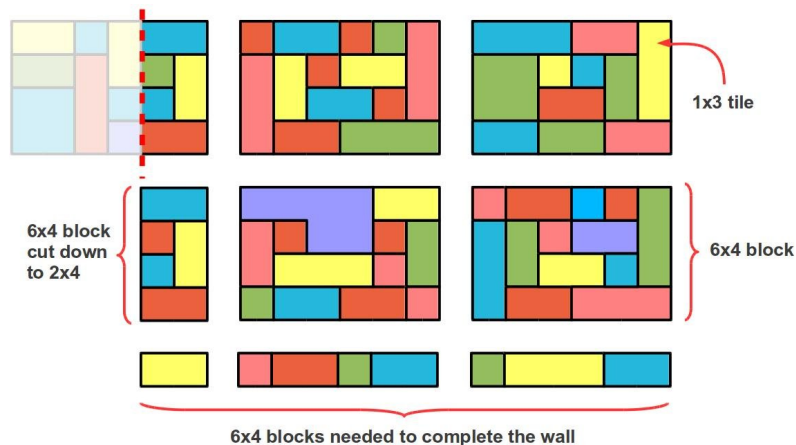


Figure 2: The tiles are all part of larger rectangular blocks

Task

Consider a wall that is apparently made of several small tiles. These tiles are not necessarily rectangles and can be imagined as a group of squares joined together at their edges.

A tile can only belong to one block. The wall is in fact made of larger rectangular blocks always having the same dimensions and orientation (but not necessarily the same tile patterns).

Near the wall edges, blocks can be cut into smaller pieces to fit the available space (but always maintaining the same orientation). Inner blocks, however, will always have the same size.

Given the configuration of a wall, as a grid, where a tile is defined as a collection of adjacent grid squares having the same capital letter, determine the smallest block that could be used to create the desired illusion.

Input

The first line of the input will contain two integers, **W** and **H**, representing the size of the wall. The next **H** lines will contain **W** characters from 'A' to 'Z' representing the wall's configuration. If two consecutive characters (horizontally or vertically) on the grid are the same then they belong to the same tile.

Output

A line with two integers representing the width and height of the smallest block that could be used.

Constraints

$1 \leq \mathbf{W}, \mathbf{H} \leq 1,200$ width and height of the wall

Input example 1

```
14 9
AABAABCEAAAEED
CDEBDDECCDACD
ADEDAABECCBBCD
BBEBBCCCAACCEE
AAFFFFDDEBBABC
BDEBFFBCACEFFC
ADEDDDECACDDAC
BBCAABBEABBEED
DDEBBCAACDDDA
```

Output example 1

```
6 4
```

Input example 2

```
11 11
ABBABBBBAABD
BCCBCCDEBAC
BCABDDDEBAC
ABBABABAABA
BAABDDDBDAB
BCACBBDBDAC
BDABDDDBBAD
ABCACACAACE
ACCBAAABACA
BAACDDDABAB
ACCAEEZZYX
```

Output example 2

```
3 3
```

Problem G: The Open Day

Once per year, your University plans the visit of high school students in a single day. Each department at the University proposes a set of events. Someone from the central services will have to schedule these events in a large open space. Events can be scheduled simultaneously as long as no high school student has signed for two of them.

At 14:00, the prime minister plans to visit some of the events that are running at that time. For this reason, the rector of the University would like to schedule as many events as possible for that timeslot.

This looks a hard problem for the central services. So, they decided to call you asking for your help.



Task

The central services told you the number N of events that can be scheduled at 14:00, and which M pairs of events cannot run simultaneously because they have high school students in common. Your task is to write a program that computes the maximum number of events that can run simultaneously without having students in common.

Input

Each test case starts with two positive integers: N and M . Then, M lines follow. Each line contains the index of two events that cannot run simultaneously because they have at least one student in common. The indices of the events are from 0 to $N - 1$.

Output

A single line containing the maximum number of events that can run simultaneously.

Constraints

$1 < N \leq 30$ Number of events

$1 < M \leq 435$ Number of pairs of events that cannot run simultaneously

Input example 1

```
6 6
0 1
0 2
0 3
0 4
0 5
1 5
```

Output example 1

```
4
```

Input example 2

```
8 4
3 1
1 4
0 2
1 0
```

Output example 2

```
6
```

(this page is intentionally left blank)

Problem H: Caesar's Orders

Julius Caesar uses a cipher to conceal from prying eyes the orders that he sends to his generals. This cipher, known as Caesar's cipher, is a simple three left shift substitution cipher: each character in the original message is substituted, in the ciphered message, by the character which is three positions to the right in the alphabet. For example: A, in the original message, is substituted by a D in the ciphered message, B by E, etc, and Z by C.

Lately, Julius Caesar is growing suspicious about the secrecy of his cipher and decided to make it more secure. The new process to encipher messages uses the good old Caesar's cipher, plus steganography (steganography is the art and science of hiding messages within messages in such a way that nobody even suspects that the hidden messages are there) and also an elaborated alphanumeric to numeric substitution.

The process of ciphering a message has four steps. It starts with **C** - the Caesar's order - and **M** - a message in which the order will be hidden. All the messages use capital letters only. Let N_C be the number of characters in **C** and N_M be the number of characters in **M**.

For example, suppose **C** is "HOLDYOURPOSITION" and **M** is "ABCDEFGHIJKLMNOPQRSTUVWXYZ" (despite the example, **M** does not have to contain all the letters of the alphabet, but only all the letters in the message). N_C is 16 and N_M is 26.

1. A Caesar's cipher is applied to **C**, to obtain the ciphered order **C**₂, with a left shift of¹ $\lfloor \frac{N_C}{3} \rfloor$
Step example: With a left shift of 5 we obtain **C**₂ = MTQIDTZWUTXNYNTS.
2. Each character of **C**₂ is mapped to a position of **M**, obtaining an integer array **L**.
Step example: **L** = 12 19 16 8 3 19 25 22 20 19 23 13 24 13 19 18.
3. The Caesar's cipher is applied again, now to **M**, with a left shift of² $\lceil \frac{N_C}{2} \rceil$, producing the ciphered message **M**₂.
Step example: **M**₂ = IJKLMNOPQRSTUVWXYZABCDEFGH.
4. The final ciphered message **M**₃ is obtained. This is the message that will be sent to the generals. In order to build **M**₃, each character of **M**₂ is converted to a number as follows: A is mapped to 00; B is mapped to 01; ...; Z is mapped to 25.
Step example: **M**₃ = 0809101112131415161718192021222324250001020304050607

Task

Your task as Roman master spy allocated to a general is to decipher the message **M**₃ received from Julius Caesar and deliver the order **C** to your boss, the general.

Input

The first line of the input contains the number N_C of letters in the order issued by Julius Caesar. The second line contains a set of N_C numbers separated by single space that constitute the integer array **L**. The last line contains the ciphered message **M**₃.

Output

The output consists of a single line with the order **C** given by Julius Caesar.

¹ $\lfloor x \rfloor$ represents the greatest integer less than or equal to x .

² $\lceil x \rceil$ represents the smallest integer greater than or equal to x



Constraints

$1 \leq N_C < 1,000$ Number of characters of the order
 $1 \leq N_M < 15,000$ Number of characters of the message

Input example 1

```
16
12 19 16 8 3 19 25 22 20 19 23 13 24 13 19 18
0809101112131415161718192021222324250001020304050607
```

Output example 1

HOLDYOURPOSITION

Input example 2

```
10
19 4 159 69 43 13 5 10 60 9
241209101619192205180807091316131811102518072413191805220925232505161603240320092309
240113241216091024051808221311122423212505220906220507150924230112092209241209252020
092210192210161919221025180724131918192216190109221019220709131613181110251807241319
181219221304191824051606052223052209171323231318110603220517051825140518
```

Output example 2

RIGHTFLANK

Note: the second input example has only three lines. The 3rd and last line, containing \mathbf{M}_3 , is shown on paper in different lines only because of space constraints. You can however be assured that all input files will contain exactly 3 lines

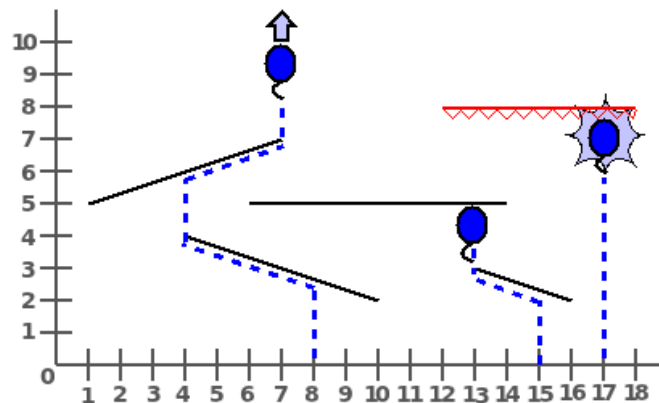
Problem I: Flying Balloons

Balloons are intimately connect to competitive programming. On contests such as MIUP, for each problem solved you get a balloon with the color associated to the respective problem. After the closing ceremony, it is a tradition to just let the balloons go and fly. Some of the balloons get stuck on some parts of the ceiling, others blow up on spiky surfaces and others just wander into the open air. We need to predict the trajectory of each balloon.



Balloons are filled up with helium and will always fly up in a straight vertical line, unless they encounter some obstacle. You are given a 2D view of the part of the world you need to consider. For simplification, the balloon may be considered as a single point with almost negligible size. Obstacles are given as line segments and may be of two types: spiky and normal. When a balloon touches a spiky obstacle, it immediately blows up regardless of its slope. When it encounters a normal obstacle, it may do one of two things. If the obstacle is an horizontal line it will get stuck on it. If it has a slope, no matter how small it is, it will continue flying, turning to the right or to the left depending on the angle of the slope.

The following figure illustrates an example world with five obstacles (one of them is spiky), and what would happen with three different ground locations, given by the X coordinate. The balloon launched on $X = 8$ will fly into the open space after touching two obstacles. The balloon of $X = 15$ will get stuck on the 2nd obstacle it encounters. Finally, the balloon launched on $X = 17$ will be blown up on the first obstacle it will collide with, because the obstacle is spiky.



Task

Given a set of L line segments, identified by its endpoints, and a set of N balloon ground launch coordinates, your task is to compute the trajectory of all the balloons, identifying whether they will wander into the open space, get stuck or get blown up. You must also calculate how many obstacles they will collide with, before fulfilling their destiny.

Input

The first line contains a single integer L , indicating the number of line segments. Then follow exactly L lines, each one in the form $X1_i \ Y1_i \ X2_i \ Y2_i \ C$, describing the i -th line segment. Each line segment has its endpoints at the integer coordinates $(X1_i, Y1_i)$ and $(X2_i, Y2_i)$ and C is a single character, being 'N' if the line segment is normal, and 'S' if it is spiky. Both the segments and their endpoints can be in any order.

After this comes a single integer \mathbf{N} , indicating the number of balloons to consider. Then come \mathbf{N} lines, each one with a single integer \mathbf{B}_i indicating the X coordinate of the i -th balloon. You can assume that all balloons will be launched at ground level, that is, with $Y = 0$.

You can also assume that no two X coordinates will be the same, regardless of being from an endpoint or a balloon, and that there are no intersections between line segments.

Output

The output should be composed of \mathbf{N} lines, one line per balloon in the same order they appeared in the input, saying what happened to each balloon. Each line may be of one of the following three types:

- **fly** S , indicating that the balloon goes into the open space after hitting S segments;
- **stuck** S , indicating that the balloon gets stuck after hitting S segments (the S -th obstacle is horizontal and the balloon stays there).
- **blow** S , indicating that the balloon blows up after hitting S segments (the S -th obstacle is spiky and the balloon blows up there).

Constraints

$1 \leq \mathbf{L} \leq 100$	Number of line segments
$1 \leq \mathbf{X1}_i, \mathbf{Y1}_i, \mathbf{X2}_i, \mathbf{Y2}_i \leq 40,000$	Coordinates of line segment endpoints
$1 \leq \mathbf{N} \leq 100$	Number of balloons
$1 \leq \mathbf{B}_i \leq 40,000$	X Coordinates of balloon launching points

Input example

```
5
4 4 10 2 N
16 2 13 3 N
6 5 14 5 N
12 8 18 8 S
7 7 1 5 N
6
8
15
17
19
2
11
```

Output example

```
fly 2
stuck 2
blow 1
fly 0
fly 1
stuck 1
```

Explanation: *The line segments of the sample input, and the first three balloons, are the ones depicted in the given figure. The balloon launched on $X = 19$ flies away without touching any object. The balloon launched on $X = 2$ touches one obstacle and then flies away. Finally, the balloon of $X = 11$ gets stuck on the first segment it encounters, which is horizontal.*