New Zealand Programming Contest 2018

PROBLEM SET

PREAMBLE

Please note the following very important details relating to input and output:

- Read all input from the keyboard, i.e. use `stdin`, `System.in`, `cin` or equivalent. Input will be redirected from a file to form the input to your submission.

- Do NOT prompt for input as this will appear in your output and cause a submission to be judged as wrong.

- Write all output to the screen, i.e. use `stdout`, `System.out`, `cout` or equivalent. Do not write to `stderr`. Do NOT use, or even include, any module that allows direct manipulation of the screen, such as `conio`, `Crt` or anything similar.

- Output from your program is redirected to a file for later checking. Use of direct I/O means that such output is not redirected and hence cannot be checked. This could mean that a correct program is rejected! You have been warned.

- Unless otherwise stated, all *integers* will fit into a standard 32-bit computer word. If more than one integer appears on a line, they will be separated by spaces.

- An *uppercase letter* is a character in the sequence 'A' to 'Z'. A *lower case letter* is a character in the sequence 'a' to 'z'. A *letter* is either a lower case letter or an upper case letter.

- Unless otherwise stated, a *word* or a *name* is a continuous sequence of letters.

- Unless otherwise stated, a *string* is a continuous sequence of visible characters.

- Unless otherwise stated, words and names will contain no more than 60 characters, and strings will contain no more than 250 characters.

- If it is stated that ‘a line contains no more than *n* characters’, this does not include the character(s) specifying the end of line.

- Input files are often terminated by a ‘sentinel’ line, followed by an end of file marker. This line should not be processed.

PROBLEM A BUS SERVICE 3 POINTS

A small rural bus service wants some data on passenger usage of some of their less popular routes. They want to be able to use appropriately sized buses and to set the frequency of the service to meet demand.

From the data provided, you are to work out the total number of passengers carried on the service, and the highest number of passengers on board at any time.

Input

The first line of input contains the route number consisting of up to 5 characters (which may include letters).

The next line contains a single non-negative integer, the number of passengers on the bus when it departed.

Remaining lines of input consist of two non-negative integers, separated by a space. This represents the number of passengers who got on, and the number who got off at a stop, in that order. The last line will contain 0 0. This line should not be processed.

None of the numeric values will be higher than 80.

Output

Three lines of output are required as follows:

```
Route <route number>
Total <passenger total>
Highest <passenger highest>
```

where

`<route number>` is the route number read from the input.

`<passenger total>` is the total number of passengers who got on the bus.

`<passenger highest>` is the maximum number of passengers on the bus at one time. Assume that all alighting passengers get off the bus before any get on.

Turn over for sample input and output.

Sample Input

```
35A
5
3 0
4 2
2 1
0 1
5 2
6 0
3 6
0 0
```

Output for Sample Input

```
Route 35A
Total 28
Highest 19
```

PROBLEM B **ORDERING HER TOYS** **3 POINTS**

Natalie was a little girl who loved to organise her toys. Her favourite game was to line up her toys in alphabetical order by their name then pair them up the first with the last, the second with the second to last etc.

For example, one set of toys was an ant, a bee, a yak and a zebra. She paired Ant and Zebra, Bee and Yak.

Your problem is to assist Jane in ordering toys in the same way when she goes to a friend's house – she just loves order so much!

Given a list of toys, order them so that the one with the name which comes first alphabetically is put with the one which comes last alphabetically, the second with the second last and so on until all the toys have been paired. If there is an odd number of toys, the one in the middle must be displayed on its own after all the pairs.

Input

The first line of input will be a positive integer, T, which is not greater than 20. This tells you how many toys there are at this friend's house. Then there are T lines each listing a toy.

Toy names will not be more than 12 characters long, may contain spaces and will all begin with a capital letter. They will be not be listed in any particular order.

Output

Output should consist of the pairs of toys, each pair on a separate line. The first in each pair should be the toy with the name which comes first alphabetically. The pair should be separated by a comma. If there is an odd number of toys, the last listed should not have a comma following it.

The pairs must be listed in alphabetical order of the first toy of the pair.

Turn over for sample input and output

Sample Input 1

```
4
Bee
Zebra
Ant
Yak
```

Output for Sample Input 1

```
Ant, Zebra
Bee, Yak
```

Sample Input 2

```
5
Yacht
Doll
Cat
Car
Lego
```

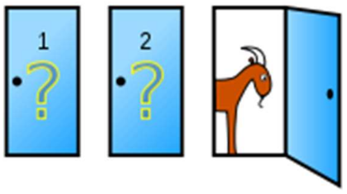
Output for Sample Input 2

```
Car, Yacht
Cat, Lego
Doll
```

PROBLEM C MONTY HALL 3 POINTS

The Monty Hall problem is explained on Wikipedia as follows:

Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?



You win whatever is behind the door you choose. Probability theory says that to win the car your best chance is to switch from your original choice, a theory you are to investigate in this problem.

Input

The first line of input is a single integer, N (2 <= N <= 10), the number of games which follow. Each game consists of a single line containing three characters, separated by spaces.

The first character is the door chosen by the contestant, 1, 2 or 3.

The second character is the door behind which is the car, also 1, 2 or 3.

The third character is S or R. S means the user chose to switch to the other unopened door, R that the user remains with the original door. In the set of games, there will be at least one switch and at least one remain.

Output

Does the user have a better record (% success) switching doors or remaining with the door originally chosen?

If switching proves the better option, output `Switching is better.`

If not switching proves the better option, output `Switching is worse.`

If neither option proves better, output `No preference.`

Turn over for sample input and output.

Sample Input 1	**Explanation**
`6`	
`1 1 S`	`Player switches and gets a goat.`
`1 1 R`	`Player does not switch and wins the car.`
`2 1 S`	`Player switches and wins the car.`
`2 1 R`	`Player does not switch and gets a goat.`
`3 1 S`	`Player switches and wins the car.`
`3 1 R`	`Player does not switch and gets a goat.`
	`Switching wins the car 2 out of 3 times (67%).`
	`Not switching wins the car 1 of 3 times (33%).`

Output for Sample Input 1

```
Switching is better.
```

Sample Input 2	**Explanation**
`4`	
`2 2 S`	`Player switches and gets a goat.`
`1 1 R`	`Player does not switch and wins the car.`
`2 2 R`	`Player does not switch and wins the car.`
`2 3 R`	`Player does not switch and gets a goat.`
	`Switching wins the car 0 out of 1 times (0%).`
	`Not switching wins the car 2 of 3 times (67%).`

Output for Sample Input 2

```
Switching is worse.
```

PROBLEM D DATES 3 POINTS

Today is September 15th 2018. Some events for 2018 have come and gone, some (like the NZPC) are today, others are still to come. All you have to do for this problem is to take a named event with the date it occurs in 2018, and state whether it has gone, is today, or is to come.

Input

The first line of input contains a single positive integer, N, not greater than 50. There then follow N lines each representing a single event for 2018.

Each line starts with a number, followed by a space, followed by the full name of a month. The month starts with an upper case letter, with the rest being lower case. The number and name together form a valid date from 2018. This is followed by a space and the name of an event. The event consists of one or more words with a maximum of 50 characters.

Output

There is to be one line of output for each line of input, in the same order as the input. The line starts with the name of the event as shown in the input. This is followed by whichever one of these three phrases is appropriate:

```
 has gone.
 is today.
 is still to come.
```

Sample Input

```
3
15 September NZPC
25 December Christmas Day
1 April Easter Sunday
```

Output for Sample Input

```
NZPC is today.
Christmas Day is still to come.
Easter Sunday has gone.
```

PROBLEM E **COOKING** **10 POINTS**

New Zealand Loves to Cook is the latest in the never ending series of cooking shows that appear on TV these days. Each contestant is judged on three elements, entrée, main and desert. Scores are added to gives the contestants a total for the show. A special feature of this show is that the judges decide on a bonus round, and whoever wins that round scores an extra 5 points. If there is a tie for that element, all tied contestants get the bonus.

At the end of each show, the contestant with the lowest score is eliminated – sent home – and the one with the highest score gets immunity – they cannot be sent home the following show. As with many such shows, the judges make sure that there are no ties for first or last place.

Input

The first line of input will be a single integer, C, (4 <= C <= 8) which is the number of contestants. There then follow C lines, each giving the name of a contestant. Names will be not more than 8 characters in length.

There will then be 3 x C lines, representing scores awarded to each contestant for each element of the contest. Each line will have the following format:

```
<name> <element> <score>
```

`<name>` will be the name of one of the contestants.

`<element>` will be one of the upper case letters E (entrée), M (main) or D (desert).

`<score>` will be an integer between 0 and 10 inclusive representing the named contestant's score for that element.

Each contestant will have one score for each element.

The final line will be a single upper case letter (E, M or D) indicating which round is the bonus round.

Output

Output will be in the format

```
<name1> gets immunity.
<name2> goes home.
```

<name1> is the name of the contestant with the highest score

<name2> is the name of the contestant with the lowest score

Sample Input

```
4
Tom
Dick
Harry
Sally
Tom M 8
Dick D 7
Harry M 9
Dick E 8
Sally D 8
Harry E 8
Sally M 8
Tom E 6
Tom D 6
Sally E 9
Harry D 9
Dick M 7
M
```

Output for Sample Input

```
Harry gets immunity.

Tom goes home.
```

Explanation

```
      E  M  D  Total
Tom   6  8  6    20
Dick  8  7  7    22
Harry 8  9  9    26 + 5 bonus =31
Sally 9  8  8    25
```

PROBLEM F **PRODUCT DELAYS** **10 POINTS**

Happy Valley Marketing are doing some internal research to find out if they are keeping up with customer demand; they are concerned that some of their suppliers are not up to scratch, and are worried this may be upsetting too many customers. They have decided to record the number of days a customer would have to wait for each product along with its listed price. This will enable them to get a feel for how many discontented customers they have.

Input

Input will consist of data for a single day. The first line contains a number N (*0* < *N* <= *20*) which represents the number of products that were ordered on the day in question.

The product count is followed by 2 x N lines with each of the products listed. The first line for each product contains its name, which is limited to 20 characters, but may include spaces. This is followed on a second line by two non negative integers, the cost of the item and the days to wait for it to be shipped to the purchaser. Each of these fields is separated by a single space. No product costs more than $1000 and no product will be delayed by more than 100 days.

These 2 x N lines are followed by a line with the number of customers to be processed, C (*0* < *C* <= *50*). For each customer, there follows a line with the customer id (which may contain letters as well as numbers), the number of products wishing to be purchased, P (1 <= P <= 20), and the maximum days they are prepared to wait for a product, D (0 <= D <= 100). Each of these 3 items is separated by a single space. There follows P lines listing the products they wish to order, each on a single line.

Output

For each customer in the record, output the customer number followed by the value of their purchases, separated by a space, on a single line. If a product is not available within the timeframe set by the customer, then it is deemed that the product has not been bought. However, on the same line, following the value, output an asterisk (*) if that customer has not purchased something they wanted due to shipping delays. The value and asterisk, if applicable, should be separated by a single space. The final line of the day's output contains the text

"`Number of discontented customers is: `"

followed by the number of customers in that scenario who could not obtain all their products (ie the number of asterisks).

Turn over for sample input and output.

Sample Input

```
3
iPod Nano
255 0
Ducks Feet Perfume
120 15
Silver Charm Pendant
180 5
3
1001 2 5
iPod Nano
Silver Charm Pendant
1860 2 3
Ducks Feet Perfume
Silver Charm Pendant
1025 2 6
iPod Nano
Ducks Feet Perfume
```

Output for Sample Input

```
1001 435
1860 0 *
1025 255 *
Number of discontented customers is: 2
```

Explanation :

The 2 products 1001 wants can be shipped within the customer's time frame.

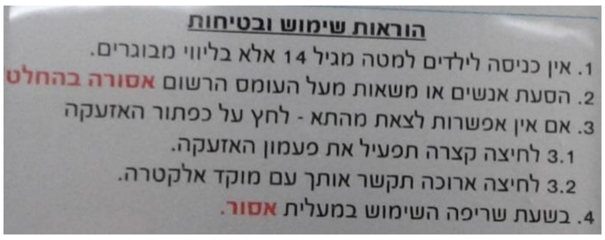
Neither of customer 1860's purchases can be shipped within his timeframe and so no purchase is made.

With customer 1025, one product can be shipped in an acceptable time, the other cannot. The iPod is thus sold (cost 255) but the perfume is not, hence the *.

This makes 2 customers who could not complete their purchases.

PROBLEM G **LETTERS AND NUMBERS** **10 POINTS**

A colleague took this photograph in the lift of a middle eastern hotel. She is a mathematician and was intrigued to see that the letters were to be read right to left (she understands the language), but that the digits are read left to right. In the second line, for example, the number 14 is fourteen not forty one.


הוראות שימוש ובטיחות
1. אין כניסה לילדים למטה מגיל 14 אלא בליווי מבוגרים.
2. הסעת אנשים או משאות מעל העומס הרשום אסורה בהחלט
3. אם אין אפשרות לצאת מהתא - לחץ על כפתור האזעקה
3.1 לחיצה קצרה תפעיל את פעמון האזעקה.
3.2 לחיצה ארוכה תקשר אותך עם מוקד אלקטרה.
4. בשעת שריפה השימוש במעלית אסור.

We both wondered what our English text would look like if we followed the same rules. For example:

1. Any person under the age of 14 must be supervised.

would be written as:

.desivrepus eb tsum 14 fo ega eht rednu nosrep ynA .1

This problem will give you some standard English sentences to which you must apply the rules described.

Input

Input will consist of a number of lines of text, each containing at least 1 character that is not white space, but no more than 100 characters. The last line will contain just the # character – this line should not be processed.

Output

The input line of text must be output so that the characters are read from right to left except for numbers. As shown in the sample, prefixes and suffixes are not counted as part of a number. Thousands separators, plus or minus signs and decimal points are, however, as long as they are not separated from the number by a space. As you can see in the photograph 3.1 is three point one, not one point three.

Sample Input

```
1. Any person under the age of 14 must be supervised.
Christmas Day is December 25th.
Cost is $18.
25 + -6 = 19
#
```

Output for Sample Input

```
.desivrepus eb tsum 14 fo ega eht rednu nosrep ynA .1
.ht25 rebmeceD si yaD samtsirhC
.18$ si tsoC
19 = -6 + 25
```

PROBLEM H **GC TOURNAMENT** **10 POINTS**

In the *NZ Stars Golf Croquet Tournament*, each competitor plays one game against each other competitor. The first to win 7 hoops wins the game. The tournament is won by the player who has won most games. Ties are settled by the higher points difference – hoops the player has won minus hoops the player's opponents have won.

Input

The first line of input will be a single integer, N, (2 <= N <= 12) which is the number of competitors. There then follow N lines, one for each competitor.

A competitor's line begins with the person's name, which is no more than 12 characters in length. There then follows a score for each game played (an integer between 0 and 7 inclusive) and an X. These are separated by spaces.

The score in the first column represents the player's score against the player from the first row, in the second column the score against the player in the second row and so on. An X is placed to mark where the column and row numbers are the same – indicating that no player has a score against themselves.

Output

Output will be in the format

```
<names> won with <w> wins and [+|-]<d> points difference.
```

<names> is the name of the winning player or players. If more than one player has an equal score, all will be listed, separated by spaces, in the order they appear in the input.

<w> is the number of games won by the player or players named.

[+|-] is + for a positive difference, - for a negative difference but no sign for a zero difference.

<d> is the points difference of the player or players named.

Turn over for sample input and output

Sample Input 1

```
4
Andrew X 7 7 4
Brenda 5 X 7 7
Colin 4 5 X 6
Daisy 7 6 7 X
```

Output for Sample Input 1

```
Daisy won with 2 wins and +3 points difference.
```

Explanation

Daisy beat Andrew 7-4, beat Colin 7-6 and lost to Brenda 6-7. She thus had 2 wins with 20 points scored and 17 points against.

Sample Input 2

```
3
Xerxes X 7 6
Yang 6 X 7
Ziggy 7 6 X
```

Output for Sample Input 2

```
Xerxes Yang Ziggy won with 1 wins and 0 points difference.
```

Explanation

Xerxes 7 Yang 6, Xerxes 6 Ziggy 7, Yang 7 Ziggy 6

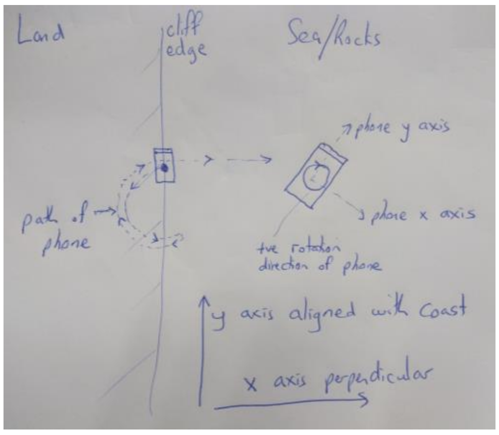
All 3 players won 1 game scoring 13 and conceding 13.

PROBLEM I — PHONE FLING — 30 POINTS

The smart phone has revolutionised modern life. Rarely do people venture from their homes now without their phones. However, the ubiquitous nature of phone usage doesn't please everyone. In particular the society of Phone Flingers has as its goal, the destruction of as many smart phones as possible. They hold meetings on a cliff top, above a rough and rocky sea. Attendees are asked to fling their phones out from the cliff. To encourage participation there is a large prize for the best fling. The prize is free registration for one team at an ACM programming contest. Of course, with the great prize on offer, it is necessary to be able to measure the throw accurately. To achieve the measurement, each participant is required to load one last app (appropriately called TheLastApp) onto their phone. The app wirelessly transmits data from the phone's gyroscope and accelerometer for two seconds after the command to throw is given.

Your task is to write a program to read a log of a phone's transmission and work out the distance from the cliff top reached at the end of a roughly two second period. Note that we need only consider this problem in two dimensions. The phone is flung using a Frisbee style of throw and so is launched and flies 'flat' – ie: spinning horizontally, but not tumbling. We need also not consider the height above the ground/sea. Although the phone will fall during its flight, the cliff is high enough to ensure that it will not reach the ground/sea within the two second period for which it is tracked.

The detail of the throw is as follows. Players stand on the cliff edge and hold the phone out horizontally in their left hands with the principle axis of the phone aligned with the cliff edge (aligned with the y axis in the sketch); the starting signal is given and the phone begins transmitting data; the player swings the phone (horizontally) to their left (anti-clockwise in the diagram), stops and accelerates the phone back clockwise, trying to let go just as the phone passes the edge; the phone flies out to sea; after about two seconds the data transmission ends; the phone crashes on the rocks or is lost in the sea. Your software is to determine the distance from the cliff edge (x coordinate of the centre of the phone) reached at the time of the last transmission as calculated from a log of the phone's messages. The phone will transmit gyroscope and accelerometer data every 0.01 seconds. In the cliff axis system (the centre of) the phone starts at position (x=0, y=0). The goal is to calculate the final x coordinate of the phone centre.



Input

Input will consist of a series of log files. For each log file the first line has a value N being the number of messages received from the phone (190 < N < 210). There will then be N lines holding space separated floating point values A, X and Y where A is the angular rotation rate, and X and Y are the phone's linear acceleration in m/sec/sec of the phone centre along the phone's x and y axes respectively. Rotation rates are measured in degrees clockwise per second (see sketch). The end of input is indicated by a line with N value zero.

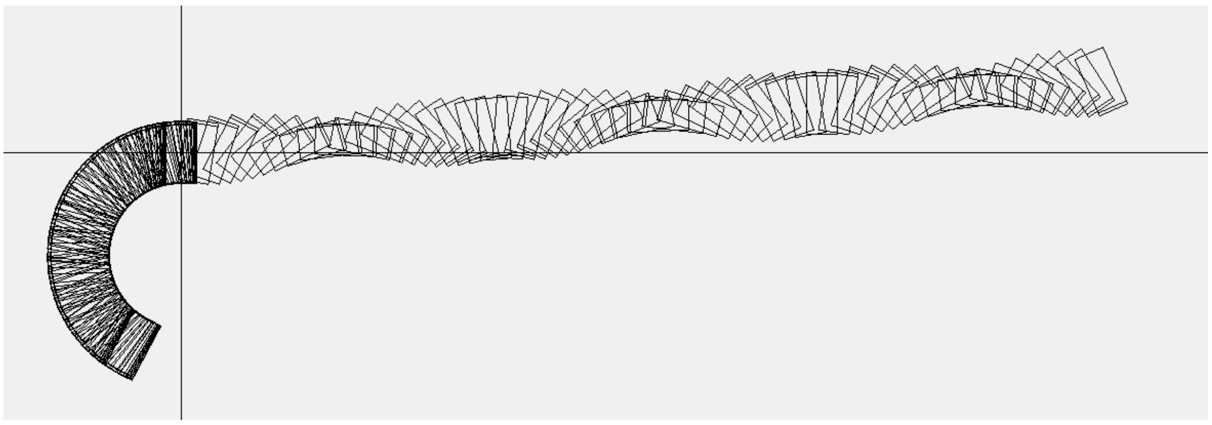
Output

For each log, output one line with the distance from the cliff edge reached (x coordinate of final phone position) in metres, rounded to one decimal place. In making your calculation the log values given can be taken to be the values applied over the 0.01 second period involved. In particular you may assume that the rotation rate given is applied for 0.01 seconds and that the acceleration values apply for the whole 0.01 seconds and are reported in the coordinates implied by the phone angle achieved at the end of the 0.01 sec time period. Problem input will be such that results are not near rounding boundaries. Note also that the input data is quite 'jittery' – the players are nervous and their backswings can be irregular.

Sample Input

The sample input is a single log trace. A few lines are printed here. The full sample input is available digitally. To help you there is a graph below showing the movement of the phone in the sample.

```
201
0 0 0
...
0 0 0
-28.28427 -49.36517 0.1221
0 49.36517 -0.1221
0 0 0
-16.43709 -28.6881 0.04093337
...
921.6003 0 0
921.6003 0 0
921.5973 0 0
0
```



Output for Sample Input

```
9.0
```

PROBLEM J **COMPARE** **30 POINTS**

Plagiarism is presenting someone else's work as your own. A method of catching culprits often used when marking assignment work is to compare the assignment text against a library of text collected from other students' work and more widely from the internet. If there is a reasonably close match then it is possible that the work was copied. The ACM sponsored company Ni Ti Nrut Ltd provides an online service for doing such checks. Of course, as soon as a way of detecting plagiarism was developed, would-be plagiarists began to develop strategies for defeating the detection.

One idea is this. Sometimes we can make letter substitutions that do not change the appearance of text very much. Such letter pairs are called 'homoglyphs'. For example, in the following sentence the letter O has been replaced by the digit 0 and the letter L has been replaced by the digit 1. Someone reading the sentence might not notice the changes, yet text comparison with the correctly typed version would see all words except 'and' as different.

```
0scar and 01ive 1earned 1atin.
```

If we move beyond Ascii and use the full Unicode character set, the number of homoglyphs becomes greater. If we were able to move further and use a full intergalactic character set it seems likely that there would be close homoglyphs for all letters. The research department of Ni Ti Nrut has asked you to develop a program to check to see what effect letter substitution might have on the comparison of sentences in English. In particular they want to know which letter substitutions have the greatest effect on matching scores.

Your task is to see how great an effect systematically substituting a single letter has on matching scores. Consider the following text (taken from Wiki).

The ACM International Collegiate Programming Contest is an annual multi-tiered competitive programming competition among the universities of world. Headquartered at Baylor University, directed by ICPC Executive Director and Baylor Professor Dr. William B. Poucher, the ICPC operates autonomous regional contests covering six continents culminating in a global World Finals every year.

A plagiarist might reword this text a little and use it in their own work. For example:

The Association for Computing Machinery's Collegiate Programming Contest is an annual multi-level programming competition between the universities of world. Its Headquarters is at Baylor University. Directed by ICPC Executive Director and Baylor Professor Dr. William B. Poucher, the ICPC operates independent regional contests covering most continents culminating in an annual global World Finals.

Ni Ti Nrut use a string matching algorithm which finds the minimum number of edit operations which can transform one text into the other. Imagine finding a letter in an alphabet from Alpha Centauri that looked like an English 'a' and substituting that for each of the a's in the plagiarised text. How would that alter the number of edit operations calculated by Ni Ti Nrut's algorithm? If you had access to the algorithm you could find out by choosing a character not in use in either text, using that to replace the a's and running the algorithm. For example, using '@'

The Associ@tion for Computing M@chinery's Collegi@te Progr@mming Contest is @n @nnu@l multi-level progr@mming . . .

Unfortunately you don't have access to Ni Ti Nrut's algorithm, so you will have to write your own in order to do the experiments.

First line: number of input sets

For each set:

First line: first sentence

Second line: second sentence

Input

Input will consist of a series of tests. The first line of the input holds a single integer N, being the number of tests (1 <= N <= 20). Then for each test, there are three lines of input. The first line holds a single letter or digit, C. The second and third lines are samples of text S1 and S2 (each of length no greater than 500 characters.

Output

For each test your program should first compute the minimum edit distance (d1) between S1 and S2. The minimum edit distance is the smallest number of edit operations that can convert S1 into S2. The permissible edit operations are: delete a character, add a character, substitute a single character for another single character. Your program should then replace all instances of the character C in S2 with the '@' character and compute the minimum edit distance (d2) between S1 and the modified S2. ('@' will not occur in the S1 and S2 input strings). Generate one output line for the test with the text "Differences d1 and d2" as shown in the sample output.

Sample Input

```
2
e
Test sentence
Tester sentence
a
The ACM International Collegiate Programming Contest
The Association for Computing Machinery's Collegiate Programming Contest
```

Output for Sample Input

```
Differences 2 and 6
Differences 31 and 33
```

PROBLEM K FOILED (30 SECONDS) 30 POINTS

NZPC Software Consultants Inc. has become a victim of its own success. In the early days of the company, to keep the programmers on their algorithmic toes, the CTO decided to pay them in gold foil. The idea was that payday started with a rectangle of gold foil. The foil was ruled into 1cm squares; it was then repeatedly folded horizontally and vertically, along the ruled lines. Programmers were then allowed to select squares of the folded foil. When a square was selected by a programmer it was cut out through the whole depth of the fold and given to them. A good choice led to the programmer receiving many squares of gold; a poor choice … not so much.

As the company grew, the system became impractical and was abandoned. But … this led to a reduction in staff algorithmic skill. The current CTO has decided to reintroduce the old system, but this time to work with a simulation of foil folding because at the current size of the company the foil rectangle size needs to be measured in kilometres. Your task is to write the folding simulator and report on the square count of cells. Folds are allowed on each side (top, bottom, left and right).

For example, consider a 3 cm by 3 cm piece of foil. At the start there is just one layer of foil on each square.

1	1	1
1	1	1
1	1	1

Fold the left edge over two cm.

2	1
2	1
2	1

Fold the bottom edge up one cm.

2	1
4	2

The fold leaves two rows and two columns. The square at row 2, column 1 has four layers of foil.

Input

Input will consist of a series of rectangles to analyse. For each rectangle the first line will have space separated integers W, H, F and P; where W and H are the width and height of the rectangle in cm, with 1 <= W, H <= 10000000 (10 km); 1 <= F <= 20 is the number of folds to be applied; and P the number of squares to be counted with 1 <= P <= 10. This will be followed by F lines, each specifying one fold in the format (T|L|R|B)d, where d is the number of cm in from the current boundary (Top, Left, Right or Bottom) at which to make the fold. Finally there are P lines each with space separated integers R and C, being the 1 based row and column numbers of squares for which layer counts are required. Input is terminated by a line with four zeros. All folds are proper in that they reduce the size of the rectangle.

Output

One line for each square to be counted holding the number of layers at that square.

Sample Input

```
3 3 2 2
L2
B1
2 1
1 1
0 0 0 0
```

Output for Sample Input

```
4
2
```

PROBLEM L **BLOCKS AND SHADOWS** **30 POINTS**

Your task is to arrange a collection of upright rectangular blocks in a line from left to right. The sun is shining from the right at a particular angle, and the sunlight causes damage to the blocks. The goal is to maximise the number of blocks that are entirely in the shade of other blocks.

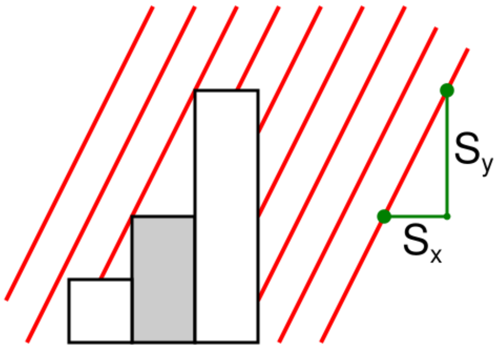


Figure 1. The middle block is in the shade.

Input

The first line will contain three integers N, S_x, and S_y ($1 \le N \le 100{,}000$, $1 \le S_x \le 10{,}000$, $0 \le S_y \le 10{,}000$). N is the number of blocks. S_x and S_y specify that the slope of the rays of sunlight is S_y / S_x.

The following line will contain N space-separated integers $H[1] .. H[N]$ specifying the heights of the blocks ($1 \le H[i] \le 100{,}000$ for $i = 1..N$). The width of each block is 1 unit.

Output

Output a single integer, the maximum number of blocks that can be arranged in shade.

Sample Input

```
3 1 2
4 2 1
```

Output for Sample Input

```
1
```

Explanation

The sample corresponds to Figure 1. There are three blocks, the slope of the sunlight is *2/1*, and the heights of the blocks are 4, 2, 1. If the blocks are arranged in the order 1, 2, 4 then the middle block is in the shade. It doesn't matter that the top left corner is almost in sunlight.

PROBLEM M **EYE OF THE TIGER (4 SEC)** **100 POINTS**

Meredith is playing SUPERHOT in virtual reality.

The game mode she is playing is unusual. She is at the end of an alleyway. Enemies run at her in single file. There is a stack of items to her left and another stack of items to her right. Meredith must kill all the enemies by throwing items at them.

The enemies must be killed in single file order since the alleyway is quite narrow and Meredith cannot accurately throw items over the heads of enemies.

The two stacks of items (left and right) available to Meredith contain various items, each of which may do a different amount of damage. Meredith does not have time to re-order the stacks or do anything other than pick up and throw a single item at a time. That is, at each point in time, Meredith only has two choices: throw the top item from the left stack (if it exists) or throw the top item from the right stack (if it exists).

A peculiarity of the game mode Meredith is playing is as follows. Each enemy has a specific number of hitpoints. To kill an enemy, Meredith must deal **exactly** the same amount of damage as they have hitpoints. If Meredith runs out of items to throw before all the enemies are dead, or does **more** damage to an enemy than their hitpoints, she loses the game. If she can kill all the enemies, she wins. Write a program to determine if Meredith will win.

Input

The input consists of four lines. The first line contains three integers E (1 <= E <= 3000), which is the number of enemies, S1 (1 <= S1 <= 3000), which is the number of items in the left stack, and S2 (1 <= S2 <= 3000), which is the number of items in the right stack.

The second line describes the enemies. This line contains E integers h1, … hE (1 <= hi <= 10^9), which are the hotpoints of the ith enemy.

The third line contains S1 integers, each between 1 and 10^9 inclusive, which are the damage amounts of items in the left stack in order from top to bottom. The fourth line contains S2 integers, each between 1 and 10^9 inclusive, which are the damage amounts of items in the right stack in order from top to bottom.

Output

If it is possible for Meredith to win, display “YES”. Otherwise, display “NO”.

Your program will be run on one problem at a time. Here are sample input and output for four problems.

Sample Input 1

```
2 2 2
4 3
3 2
2 2
```

Output for Sample Input 1

```
YES
```

Sample Input 2

```
2 2 2
4 3
3 2
2 3
```

Output for Sample Input 2

```
NO
```

Sample Input 3

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

Output for Sample Input 3

```
NO
```

Sample Input 4

```
1 1 1
2
1
1
```

Output for Sample Input 4

```
YES
```

PROBLEM N **TRAKA** **100 POINTS**

Mirko loves cars and he finally managed to start his own car factory! There are **N** workers in Mirko's factory. They are manufacturing cars on a conveyor belt, in a pipeline fashion. Workers are denoted by numbers 1 – leftmost, to **N** - rightmost. Each of the workers does his specific job and requires certain amount of time to complete it.

Production of a single car starts with worker #1 (Mirko). After he had finished with his part of the job, worker #2 takes over, after him #3... When worker #**N** finishes with his part, the car is finished. Mirko and his workers have to produce **M** cars and they **must** produce them in order 1 to **M**.

For every worker **i** we know $\mathbf{T_i}$ **-** time required for him to do his part of the job. For every car **j** we know factor of assembly complexity $\mathbf{F_j}$. Time in minutes for worker **i** to finish his part of the job on the car **j** is computed as a product $\mathbf{T_i F_j}$.

After some worker has finished working on a car, he has to give it to the next worker **instantly**, without any delay (weird company policy). For that reason, the worker receiving the car has to be free (he must not be working on some other car). In order to fulfill this condition, Mirko has to choose a good timing to start building a new car. To be efficient, he'll wait **minimum** number of minutes until he is certain that all of the conditions described are met.

Write a program which will, given worker times and factors of complexity for each car, compute total time required for producing all of the cars.

Input

First line of input contains space-separated positive integers **N** ($1 \le \mathbf{N} \le 100\,000$), number of workers, and **M** ($1 \le \mathbf{M} \le 100\,000$), number of cars.

i-th of the following **N** lines contains worker time $\mathbf{T_i}$ for the worker **i**.

j-th of the following **M** lines contains factor of complexity $\mathbf{F_j}$ for the car **j**.

These conditions hold: $1 \le \mathbf{T_i} \le 10\,000$, $1 \le \mathbf{F_j} \le 10\,000$.

Output

First and only line of output has to contain required number of minutes.

CONTINUES

Sample Tests

Input	Input	Input

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

Output	**Output**	**Output**
11	29	55

First sample description: After four minutes, first worker finishes working on the first car. He might start working on the second car immediately, but that would violate a condition that cars have to be passed to next workers as soon as they're done (after seven minutes second worker would finish working on his part of second car, but the third worker would not be free to take over as he would still be working on the first car). That is the reason production of the second car is started after five minutes. Production of the third car starts after seven minutes. First car is finished after eight, second after nine and third after eleven seconds. Total time is then 11.

PROBLEM O **SNAKES AND LADDERS** **100 POINTS**

Snakes and Ladders is a game played on a board with numbered squares. There are snakes and ladders that connect pairs of squares. The player start at square 1. To advance, the player rolls a 6-sided die and moves forward by the number of squares rolled. If the player lands on the head of a snake or the foot of a ladder, the player moves to the connected square. The game ends when the player reaches the last square or moves past it.

Given a description of the board, your task is to determine the average number of die rolls that it takes to play the game.

Input

The first line will contain a single integer *N*, the number of squares ($2 \le N \le 100$).

The following line will contain *N* space-separated integers *J[1] .. J[N]* describing the squares. If square *i* contains the head of a snake or the foot of a ladder, *J[i]* will be the number of the connected square. Otherwise *J[i]* will be 0.

It is guaranteed that $J[0] = 0$, $J[N] = 0$, and that if $J[i] \neq 0$ for some *i* then $J[J[i]] = 0$ (in other words, the snakes and the ladders do not form chains).

Output

Suppose the average number of die rolls to play the game as a simplified fraction is *p/q*. Output $p \times q^{-1}$ modulo 1000000007, where q^{-1} is the inverse of *q* (that is, an integer *b* such that $q \times b \equiv 1$ modulo 1000000007).

It is guaranteed that the player will always be able to reach the last square and that q^{-1} exists.

Sample Input 1

```
3
0 0 0
```

Output for Sample Input 1

```
166666669
```

Sample Input 2

```
5
0 1 1 1 0
```

Output for Sample Input 2

```
2
```

Sample Input 3

```
4
0 0 0 0
```

Output for Sample Input 3

```
361111115
```

Explanation

In the first example there are three squares. If the player rolls 2 or higher (probability: *5/6*) they will reach the last square on the first turn. If the player rolls 1 (probability: *1/6*) they will move to square 2 and reach the last square on the next turn. So the average number of rolls is $(5/6) \times 1 + (1/6) \times 2 = 7/6$. It can be shown that 6^{-1} modulo 1000000007 is 166666668, so the output is $7 \times 166666668 \equiv 166666669$ modulo 1000000007.

In the second example there are five squares and there are snakes on squares 2, 3 and 4 that take the player back to square 1. The player needs to roll 4 or higher to reach the last square, otherwise they will land on the head of a snake and move back to the start. The average number of rolls needed is 2.

In the third example the average number of rolls is *49/36*.

PROBLEM P **MAKING PHRASES** **100 POINTS**

Niklaus likes writing rules for making phrases. The rules involve identifiers (names) and literals (double-quoted strings). A rule states that a particular identifier may be replaced with a sequence of identifiers and literals.

Here are some example rules to illustrate:

```
clause = noun verb noun .
noun = "I" .
noun = "ice" "cream" .
verb = "like" .
```

One rule is given in each line. The identifiers used are `clause`, `noun`, and `verb`, and the literals used are `"I"`, `"ice"`, `"cream"`, and `"like"`. The first rule says that `clause` can be replaced with `noun verb noun`. The second rule says that `noun` can be replaced with the literal `"I"`, and so on.

A phrase is made by starting with some identifier and then applying rules until the result consists only of literals.

Using the rules above with `clause` as the starting identifier, the phrase `"I like ice cream"` can be produced using the following steps:

1. Start with `clause`
2. Apply rule 1 to get `noun verb noun`
3. Apply rule 2 to the first `noun` to get `"I" verb noun`
4. Apply rule 4 to `verb` to get `"I" "like" noun`
5. Apply rule 3 to `noun` to get `"I" "like" "ice" "cream"`

When there are a lot of rules it can get confusing for Niklaus, and looking at some small example phrases can be really helpful. We'll define the length of a phrase to be the number of literals in it. Given a set of rules, a starting identifier, and a target literal, your task is to find the shortest phrase containing the target literal that can be made and output its length. If no such phrase can be made, output -1. If the length of the shortest phrase is greater than 10^9, the situation is completely hopeless so output -1 as well.

Input

Input will consist of a sequence of problems. The first line for each problem will have an integer N, the number of rules ($1 \le N \le 100{,}000$). The following line will have two space-separated tokens: the starting identifier, and the target literal in double quotes. The following N lines will describe the rules, one per line as in the example above. In detail, the description of a rule will have the following space-separated tokens:

1. The identifier that can be replaced using this rule
2. An equal sign (=)
3. Zero to ten identifiers and/or double-quoted literals which form the replacement
4. A period (`.`)

Identifiers will consist of 1 to 20 lowercase and/or uppercase letters. Literals will consist of a double quote ("), 1 to 20 lowercase and/or uppercase letters, then another double quote. In particular, literals will never contain embedded spaces. Identifiers and literals are case-sensitive. Every identifier will appear on the left hand side of at least one rule.

Each problem will be followed by a blank line. End of input will be denoted by N = 0 and should not be processed.

Output

For each problem, output a single integer with the number of literals in the shortest phrase containing the target literal. If no phrase containing the target literal can be made, or if the answer is greater than 10^9, output `-1`.

Sample Input

```
4
clause "ice"
clause = noun verb noun .
noun = "I" .
noun = "ice" "cream" .
verb = "like" .

2
dinner "dessert"
dinner = spaghetti "dessert" .
spaghetti = "some" "spaghetti" "then" "more" spaghetti .

1
pantry "chocolate"
pantry = "spaghetti" .

14
rules "QUOTE"
rules = rule .
rules = rule rules .
rule = identifier "EQ" list "DOT" .
list = .
list = term list .
term = identifier .
term = literal .
identifier = letters .
literal = "QUOTE" letters "QUOTE" .
letters = letter .
letters = letter letters .
letter = "A" .
letter = "B" .
letter = "C" .

0
```

Output for Sample Input

```
4
-1
-1
6
```

Explanation

Problem 1: Starting from `clause`, the shortest phrase containing "`ice`" that can be made is "`I like ice cream`" which has length 4.

Problem 2: Dinner consists of spaghetti followed by dessert. Spaghetti consists of some spaghetti then more spaghetti, which consists of some spaghetti then more spaghetti, and so on. Dinner never ends!

Problem 3: The problem asks to find chocolate in the pantry, but there is only spaghetti.

Problem 4: One shortest phrase containing "`QUOTE`" is "`B EQ QUOTE A QUOTE DOT`" which has length 6.