12th Annual Manitoba Programming Contest
September 29, 2018

General Instructions:

1. This document is printed double-sided. Please read both sides of each page.
2. Submit solutions using the PC^2 software.
3. All input should be read from standard input.
4. All output should be written to standard output.
5. Each problem has a twenty (20) second time limit for CPU time when executed on the judging data.

UNIVERSITY OF MANITOBA

Problem 1 – Charging Station

You are responsible for charging electric cars at a pay-per-use electric car charging station. The station can charge one car at a time. Before the start of each month, electric car owners use an app on their smartphone to tell you when they plan to arrive and how large their battery is. From this, the app tells you how long the car will take to charge. You would like to arrange appointments at the charging station for the next month.

Customers can plan their arrivals on the hour or half hour, and all batteries in all cars take multiples of half an hour to charge. The app converts all half-hour increments to numbers in the range 0-10080. These numbers represent times for the month (10080= 2*24*7*30). Thus, each planned arrival is represented as two numbers: an arrival time (0-10079) and the end time (1-10080). You are ensured that no charging time will go over the end of the month.

You receive all your customers' planned arrival times and have to decide who to accommodate and who must be turned away. Given the list of customers' planned arrival information (arrival time and departure time), calculate the total amount of charging that you can do in the month. You may assume that there is no turnover time in appointments, so that if one appointment ends a certain time, the next appointment can start at that same time.

For instance, suppose the following planned arrivals have been received:

- Arrival time: 0, end time: 3
- Arrival time: 4, end time: 17
- Arrival time: 0, end time: 5
- Arrival time: 0, end time: 10
- Arrival time: 13, end time: 17

From these, we can accommodate the first two customers, and this will result in 3+13=16 units (i.e., 8 hours) of charging. This is the most that can be accommodated from these planned arrivals from customers: any other combination will yield less time charging or have an overlap between a starting time for one charging and the end time of another charging.

Input

The first line of input is a single integer n with $0 \le n \le 10000$ that gives the number of planned arrivals. The next n lines each give one customer's information as two integers x and y, in that order. These represent the arrival time and the end time, with $0 \le x < 10080$, $0 < y \le 10080$ and $x < y$.

Output

Output a single non-negative integer less than or equal to 10080 representing the maximum number of units of time that

Sample Input	Sample Output

`5`
`0 3`
`4 17`
`0 5`
`0 10`
`13 17`
	`16`

Problem 2 – Curling

Curling is a sport popular across Canada. The mechanics of scoring points in curling are not relevant to this question: all you need to know is that in curling two teams play each other in a series of ends (nine in this question) and only one team can score points (maximum eight) in each end.

There are two common ways to display a score. The first is similar to scoring used in baseball, where each end shows who scored points that end and how many. For instance, consider the following scoreboard:

End	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Total**
Team 1	0	2	0	0	1	0	0	3	3	9
Team 2	3	0	1	0	0	0	0	0	0	4

This scoreboard shows that Team 1 scored two points in the second end, one in the fifth and three in each of the eighth and ninth ends, for a total of nine points. On the other hand, team 2 scored four points: three in the first end and one in the third end.

The second way to display a curling score shows the progression of scores of each team, rather than explicit information on each end. This type of scoreboard has the advantage that only one number of each value is needed for the scoreboard (for non-electronic displays). For the same game as above, the second type of scoreboard would read:

Team 1		2	5			8			9
Score	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Team 2			1	3					

A team has a number `x` in a column labeled `i` if the team's score after the `i`-th end is `x`. For example, Team 2 has a 1 in column 3 in the table, which shows that the team scored three points in end 1. Similarly, Team 1 has an 8 in column 6, representing that after the sixth end, Team 1's score was 8.

Write a program that converts from the first scoreboard format to the second type of scoreboard. You may assume that in each game each team scores a maximum of nine points.

Input

The input consists of two rows of numbers. Each number is separated from the next by a single space. The first nine numbers (all between 0 and 8) give the scores for each of the two teams in each of the nine ends. The last number on each line is the final score for the team in that line (which is at most nine).

Output

Write three lines of output, representing the score for each team, separated by a row of numbers, as depicted above. The first row should give the score for the first team (in the same order as the input). The second row should denote the score counter, which is the numbers 1 through 9, each preceded by exactly one space (including the number 1). The third row should give the score for the second team. If a score is not used, indicate this with a single underscore (i.e., the single character "_"). All scores up to end nine should either be a digit or an underscore for both teams, and all characters in the first and third rows should be lined up with a score in the second row (including the space before the number 1 in the middle row).

Sample Input 1	Sample Output 1
`0 2 0 0 1 0 0 3 3 9`	` _ 2 5 _ _ 8 _ _ 9`
`3 0 1 0 0 0 0 0 0 4`	` 1 2 3 4 5 6 7 8 9`
	` _ _ 1 3 _ _ _ _ _`

Sample Input 2	Sample Output 2
`4 0 2 0 0 1 2 0 0 9`	` _ _ _ 1 _ 3 6 _ 7`
`0 0 0 1 1 0 0 2 1 5`	` 1 2 3 4 5 6 7 8 9`
	` 4 5 _ 8 9 _ _ _ _`

Problem 3 – Kindergarten Names

You teach kindergarten and every year, young students enroll with creatively-spelled names. Among these, there are several names that are essentially the same, but vary in spelling, like “Catherine” and “Kathryn”. You would like to group these names together.

To help you with grouping names together, you’ve decided to use the “match rating approach”, originally developed by an airline in the 1970’s to help index names that sound similar. The algorithm has two stages: encode the names, and then compare them to get a score. Higher scores indicate more similar sounding names.

To encode a word, first remove all the vowels in the word (unless the first letter is a vowel, in which case it is kept). Vowels are always A, E, I, O and U only. Then replace all double letters with a single letter. Finally, if the word is still more than six characters long, truncate it by keeping only the first three letters and the last three letters. For instance, “Catherine” is encoded as “Cthrn” and “Allyson” is encoded as “Alysn”. Upper and lower case are ignored in the encoded words.

To compare two encoded words, first we scan the words from left to right and remove all matching letters in the same positions of both words. We then repeat the same process on the words from right to left: starting at the last character in both words, delete them if they are same, and then move left one step in both words and repeat this process. After all the letters are deleted in this way, we take the number of letters remaining in the longer word, and call it n. The similarity score for the two original words is `6-n`.

Write a program that accepts two words and computes the match rating approach score for the two words.

Input

The input is two words on two lines of input. The words contain only upper and lower case letters (a-z and A-Z). There are no spaces or other symbols in the words.

Output

Output a single integer giving the match rating for the two pairs.

Sample Input 1	Sample Output 1
`Catherine` `Kathryn`	`4`

Sample Input 2	Sample Output 2
`Geoffrey` `Jefferey`	`5`

Problem 4 – Sig Figs

Significant figures (or significant digits) are the number of digits in a number that are relevant to a calculation. Tracking the number of significant figures in a computation is commonly done in engineering disciplines to ensure that the uncertainty in a calculation is tracked. In this problem, all numbers have a decimal point, so the number of significant figures of a number is the number of digits in the number that are not leading zeroes. For instance:

- 0.35 has two significant figures (3 and 5).
- 0.00004200 has four significant figures (every digit starting with 4).
- 10000. has five significant figures (all of the digits are significant).
- 1004.500 has seven significant figures.

When multiplying two numbers, the number of significant figures in the result should be the minimum of the number of significant figures in both the operands. If there are more digits in the result than significant figures, the number is rounded so that the result has the correct number of significant figures. When doing multiple operations involving significant figures, the number of significant figures F in the final answer is calculated, then for intermediate calculations, F+1 significant figures are retained. However, in the final answer, only F significant figures are used.

Write a program that takes an expression involving numbers (all of which have a decimal place) and multiplication signs, and calculate the result to the appropriate number of decimal places. For instance, given the expression 3.50*7.050, the result would be 24.7. On the other hand, given the expression 0.001*0.0005*0.0003*0.0034, the result would be 0.0000000000005.

Input

The input is a single line of text (up to 300 characters long). The only characters that appear on the line are spaces, the multiplication character (‘*’), the decimal character (‘.’) and the digits (‘0’-‘9’). In particular, there are no negative numbers or numbers in scientific notation. All numbers are valid (i.e., they have exactly one decimal point, and if they are between 0 and 1, they have a single 0 before the decimal point).

Output

Output the final expression after all multiplications have been performed, with the appropriate number of significant figures. Ensure that the output has a decimal point, even if it appears at the end of the number. If the number is between 0 and 1, place a single zero before the decimal place.

Sample Input 1	Sample Output 1
`3.50*7.050`	`24.7`

Sample Input 2	Sample Output 2
`0.001*0.0005*0.0003*0.0034`	`0.0000000000005`

Problem 5 – Plus Minus

In some team sports, like ice hockey and basketball, a statistic called *the plus-minus* of a player is calculated to give a sense of the importance of players, including players who may benefit the team but who may not score the most points. The plus-minus of a player for a particular game is calculated by adding all the points that are scored by the player's team while that player is playing and then subtracting all the points that are scored by the other team at the same time. In sports like ice hockey and basketball, where substitutions of players are common, a player is likely to not be playing when all points are scored, so different players will have different plus-minus for the same game.

For instance, if an ice hockey player is playing while three of their own team's goals are scored, but also when one of the opposing team's goals is scored, that player's plus-minus would be +2 (in ice hockey, goals are worth one point). On the other hand, if another player was not on the goal during any of the goals scored by their own team's goals, but during two goals by the opposing team, then that player's plus-minus would be -2 for that game.

You have been given the information for a game that consists of all the scoring and all the substitutions for both teams. From that information, calculate the plus minus scores for all players on both teams.

In the input, all players are identified by numbers. The two teams are called the Home (H) and Away (A) teams. All inputs will be valid: if a player is listed as entering (or leaving, respectively) the game, they will not already be in (or out, respectively) of the game.

Input

The input consists of lines of input giving information about a game. The last line of input is "`END`" on its own line, signifying the end of the game. There are two types of input lines other than the "`END`" command:

- Player change lines (`C` line): these lines give information on substitutions of players (i.e., entering or leaving the court, field or ice). They have the form "`C Z I x1 x2 .. xn O y1 y2 .. ym`" where `x1`, `x2`, … `xn` and `y1`, `y2`, … `ym` (with `n`,`m` ≥ `0`) are player numbers and `Z` is either `H` or `A` to denote the team . The `xi` players are coming **I**nto play while the `yi` players are going **O**ut of the play. The same player will not be in both the `xi` and `yi` lists.
- Scoring lines (`S` line): these have the form "`S Z a`" where `Z` is either `H` or `A` to denote the team, and a is an integer greater than zero to denote the number of points scored.

The input lines are in order in the game, so that if a player enters (via a `C` line) and a team scores (via an `S` line), then that player was playing during that score. All tokens on a line are separated from adjacent tokens by a single space.

Output

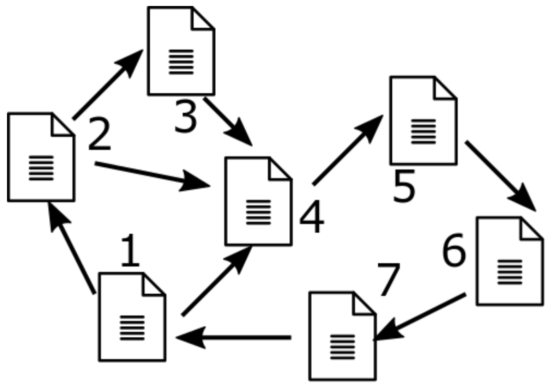
For all players mentioned in the input, list their plus-minus. List all the members of the home team first, in numerical order, followed by all the members of the away team, in numerical order. The home team should be preceded by the word “HOME” and the away team by “AWAY” as denoted in the sample output. For each player, list their number, followed by a colon and a space, and then their plus minus rating. Positive numbers and zero should have no sign. Negative numbers should have a negative sign immediately before the number.

Sample Input	Sample Output
`C H I 10 11 12 13 14 O`	`HOME`
`C A I 1 2 3 4 5 O`	`10: -1`
`S H 1`	`11: 1`
`C H I 16 17 18 19 20 O 10 11 12 13 14`	`12: -1`
`S H 1`	`13: -1`
`S A 1`	`14: -1`
`C H I 10 12 13 14 O 16 17 18 19`	`16: 0`
`S A 1`	`17: 0`
`S A 1`	`18: 0`
`END`	`19: 0`
	`20: -2`
	`AWAY`
	`1: 1`
	`2: 1`
	`3: 1`
	`4: 1`
	`5: 1`

Problem 6 – Stuck in the web

You are responsible for managing a website and are trying to track users' time on the site. You've found, through observation of users' browsing history, that people tend to stay on the site and click links as long as they don't end up at a page on the site that they have already seen (there aren't any links to any outside websites on the site).

To try and increase the amount of time users spend on the website, you would like to know how soon users could end up at a page they have already seen during the course of their clicks. For instance, suppose you have the following website:



Then the user could click five times and end up back on the same website. For instance, if the user starts at page 5, then they can click to travel to pages 6,7,1,4 and then back to 5. This is the fewest clicks that a user can make to travel back to the same page in the site, as well: any other series of clicks that takes you back to a page that you have already visited will take more than five clicks.

Given the description of a website, calculate the smallest number of clicks that can take you from a page on the website back to the same page.

Input

The input to the problem is a description of the links on the site. The first line of input gives two integers, the number of pages on the site N and the number of links on all pages L. The two integers satisfy $2 \leq N \leq 10{,}000$ and $2 \leq L \leq 50{,}000$. The pages are numbered from 1 to N. The next L lines of input all give one link per line. It is given as two integers k, j with $1 \leq k, j \leq N$ and $k \neq j$, indicating that there is a link from page k to page j.

Output

Output a single positive integer giving the minimum number of clicks that can take you from a page on the site back to the same page. The integer is guaranteed to exist: there is always a series of clicks from a page back to itself in the site.

Sample Input	Sample Output
7 9	5
1 2	
2 3	
3 4	
2 4	
1 4	
4 5	
5 6	
6 7	
7 1	