icpc international collegiate programming contest

ICPC North America Regionals 2020

ICPC Southern California Regional Contest

Official Problem Set

2020/2021 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 1
ICPC Record Matching

When using "software-as-a-service" offerings, a user often has a problem of matching records stored by the different services and determining if they refer to the same entity (person, account, order, etc.) The ICPC is no exception. While each participant has a record in the central ICPC registration system, additional "outside" applications may be used to collect and process information for functionality not provided by the central system.

Once an "outside" application is used, it becomes necessary to match the entries from both systems. Unfortunately, in spite of careful directions, sometimes it is not clear if records correspond to the same person. The primary sorts of mismatches that occur are these:

1. E-mail addresses do not match. This could be due to a misspelling, such as *.eud* instead of *.edu*, or different e-mail addresses that a participant used in the central ICPC system and the outside system.
2. Exact names do not match. This could be due to a typing error, or varying use of legal names and nicknames.

Your team is to write a program that will read lists of people from the ICPC system and an outside system and determine which records in each system do not match a record in the other system. Two entries are considered matched if *either* the e-mail addresses are an exact match *or* the last name and first name are an exact match.

Input to your program is two lists of name and e-mail address records. Each record consists of a first name, a last name, and an e-mail address, one per line, separated from each other by tabs. The first list contains the records from the central ICPC registration system. This list ends with an empty line. The second list contains the records from the outside application. The second list ends with the end-of-file. (These lists are suitable test data, not actual ICPC data.)

E-mail addresses do not exceed 64 characters in length. E-mail addresses consist of lower-case and upper-case English letters, digits, and the special characters at-sign, underscore, hyphen, and period. E-mail addresses do not begin with special characters.

First and last names do not exceed 24 characters in length. Names consist of lower-case and upper-case English letters and hyphens. Names do not begin with hyphens.

Each input list will contain at least 1 but not more than 2 000 entries. E-mail addresses and (first name, last name) pairings will be unique within each list.

Your program is to print lines showing the records from each list that could not be matched to the other list. Your program is to first print the central ICPC registration records that could not be matched, one per line. Each line should consist of the letter "`I`", a single space, the e-mail address, a single space, the last name, a single space, and the first name. These are to be printed in lexicographical e-mail address order. The e-mail addresses, last names, and first names are to be printed exactly the way they appear in the input. Once all such records are printed, the outside application records that could not be matched are to be printed the same way, except that each line should begin with the letter "`O`".

Case is to be ignored for all record matching comparisons and sorting.

Should all records from each system have a match in the other system, your program is to print a line containing only the string "`No mismatches.`".

Problem 1
ICPC Record Matching (continued)

Sample Input 1

```
Bob     Smith   bsmith@gmail.com
Randy   Nguyen  rnguyen@gmail.com
Betty   Jones   jones123@gmail.com
Sallie  Li      SalLi@aol.com

Robert  Smith   bsmith@abc.edu
Quan    Nguyen  qnguy@xyz.edu
Betty   Jones   jones123@gmail.com
Sarah   Leung   sleung@mno.edu
```

Output for Sample Input 1

```
I bsmith@gmail.com Smith Bob
I rnguyen@gmail.com Nguyen Randy
I SalLi@aol.com Li Sallie
O bsmith@abc.edu Smith Robert
O qnguy@xyz.edu Nguyen Quan
O sleung@mno.edu Leung Sarah
```

Sample Input 2

```
Camellia        Woodley Camellia_Woodley3398@deavo.com
Leroy   Thomas  Leroy_Thomas7852@fuliss.net
Freya   Campbell        Freya_Campbell9926@ubusive.com
Eduardo Allen   Eduardo_Allen3976@bauros.biz
Danny   West    Danny_West6110@twipet.com

Ed      Allen   eduardo_allen3976@bauros.biz
Daniel  West    Danny_West6110@twipet.com
camellia        woodley camellia_woodley3398@deavo.com
Leroy   Thomas  Leroy_Thomas7852@fuliss.net
freya   Campbell        campbellf@deavo.net
```

Output for Sample Input 2

```
No mismatches.
```

2020/2021 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 2
Ride-Hailing

Jamal owns a new ride-hailing business. His business operates in the following way: At least 12 hours before a trip is desired, a customer orders a trip from a starting location to an ending location to take place at a specified starting time. Although Jamal has considered ride-sharing in the past, due to the ongoing pandemic, each ordered trip is currently completed before another begins. Jamal's company has a layout of the current service area and upper bounds for the time required to drive any particular road. At some point, the team would like to incorporate traffic data to make the map dynamic, but at the moment fixed costs are what we have to work with.

By having each trip scheduled in advance, Jamal's company is able to optimize route-planning and provide an attractive business model for drivers. His company does this in the following way: Every 8 hours, Jamal selects a set of drivers to work a shift picking up and dropping off customers according to their desired schedules. Drivers are then paid per shift worked, rather than per trip completed. Jamal has found this model to be more satisfactory to drivers compared to current ride-hailing businesses. In current businesses, drivers face uncertainty as to how many rides they will be able to procure, and thus how much money they will earn. Jamal's business, on the other hand, pays drivers for every shift they work, and so a driver will either be earning money (if they work a shift) or be free to occupy their time by other means (if they aren't needed for a shift), instead of waiting around in their car hoping for a passenger to request a ride.

Unfortunately, Jamal is more of a business-type and needs help with coding for his company. In particular, given a set of ordered trips in an eight-hour window, he is lacking the algorithm to determine the minimum number of drivers that should be hired for the given shift. Can you help Jamal with this crucial task?

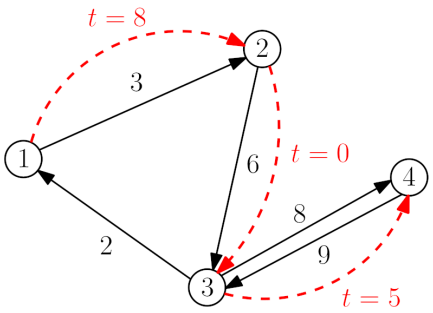


Figure 1. Illustration of sample input. Scheduled trips are depicted by dashed red lines with their respective start times. The optimal solution is to have one driver complete the trip from 2 to 3 at time 0, then travel to location 1 arriving at time 8, and then complete the trip from 1 to 2. The second driver can complete the trip from 3 to 4.

Input will begin with three integers on one line: n ($2 \leq n \leq 100$), the number of pickup or destination locations in the model of the service area, m ($1 \leq m \leq n \times (n-1)$), the number of one-directional roads in the service area, and k ($1 \leq k \leq 1000$), the number of requested trips that must be fulfilled. The next m lines each contain three integers u, v, and w ($1 \leq u, v \leq n$, $u \neq v$, $1 \leq w < 480$), indicating there is a road from location u to location v that takes w minutes to travel. Between any two locations u and v, there can be a road from u to v and from v to u but there will be at most one road in one direction between any two locations. The next k lines each contain three integers u, v, and t ($1 \leq u, v \leq n$, $u \neq v$, $0 \leq t < 480$), indicating there is a trip requested from location u to location v departing location u at minute t. There can be multiple trips requested from the same starting locations at the same time or arriving at the same ending locations at the same time. It is guaranteed every location is accessible from any other location.

Problem 2
Ride-Hailing (continued)

Output a single integer representing the minimum number of drivers that must be hired for this shift to complete all trips. Roads can be used by as many drivers as required. Assume pickup and drop-off takes 0 minutes. Picking up or dropping off at the same location does not delay trips. Assume hired drivers can drive to any starting location so they are ready to pick up any trip at minute 0 and are willing to complete trips requested before minute 480 that finish on or after minute 480. Customers must be picked up exactly at their desired pickup time.

Sample Input

```
4 5 3
1 2 3
2 3 6
3 1 2
3 4 8
4 3 9
1 2 8
2 3 0
3 4 5
```

Output for the Sample Input

```
2
```

2020/2021 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 3
Curve Speed

To help with vehicle stability, the outer edge of a road in a curve is raised with respect to the inner edge. This is called superelevation and is specified as the difference in elevation divided by the width of the road. It needs to be higher for faster speeds and sharper curves.

The radius of a curve is the radius of the section of a circle along the middle of the road where the curve is constant. See Figure 1 for a drawing of this.

In some cases the curve may need a lower speed limit than straight portions of the road. The superelevation shouldn't be more than about .12 to keep vehicles from sliding off the road in slippery conditions.

Your job is to calculate the maximum speed on a curve, given the radius of the curve and the superelevation.

The maximum speed is given by this formula:

$$V = \sqrt{(R \times (S + .16))/.067}$$

where V is the maximum speed in miles per hour, R is the radius of the curve in feet, and S is the superelevation.

The input is a series of lines terminated by end-of-file. Each line will be a test case consisting of R and S separated by whitespace. R will be an integer greater than 99 and less than 5281 and S will be a real number greater than .009 and less than 1.0. Neither will have leading zeros. There are at most 100 lines of input.

For each test case print a line containing the maximum speed rounded to the nearest integer. It is guaranteed the answer before rounding will not be within 10^{-3} of a half-integer value.

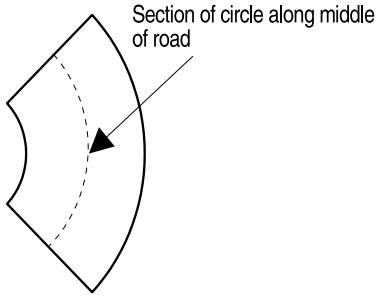


Figure 1. Section of a circle along the middle of a road with radius R.

Sample Input

```
1433   .09
1433 .12
2000 .09
600 .12
```

Problem 3
Curve Speed (continued)

Output for the Sample Input

```
73
77
86
50
```

2020/2021 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 4
LogDB

A LogDB database is filled with facts. A fact is a name with a body consisting of a parenthesized list of names. This is similar to a function call:

$fact1(arg1, arg2, arg\_3, 4, 4)$ $thing(p1, arg2, p3)$

$fact1$ is the name of the fact and the list of names in the body are the arguments of the fact. *thing* is another fact.

The names in the list are separated by commas and optional whitespace. There will be at least one name in the list. Names consist of alphanumeric characters (a–z, A–Z, 0–9) plus '_'. However, the name of a fact and the names in the body cannot start with an '_'. Names, parentheses, and commas may be preceded and followed by whitespace. However a fact or query cannot be split across lines.

Facts with different numbers of arguments are different facts. A fact may appear multiple times in the database.

Queries are like facts except the argument list can contain variables. Variables are names that start with '_'.

A query searches the database for facts with the same name as the query, that have the same number of arguments as the query, and where the names in the fact body match the names in the query body in their respective positions.

A variable consisting of only '_' is special and will match any name. A variable other than '_' will also match any name, but if that variable appears more than once in a query, the names in the fact must be the same. For example, a query:

$fact1(arg1, \_, \_, \_check, \_check)$

will match $fact1(arg1, arg2, arg\_3, 4, 4)$

Variables are only defined in the query they appear in. They are not necessarily related if they are in different queries.

The input consists of two sections: facts and queries. Judged data have no syntax errors.

The fact section consists of a series of lines of at most 200 characters. That section is terminated by a blank line. Each line consists of facts which may (or may not) be separated by whitespace. The fact section will consist of no more than 200 lines.

The query section consists of a series of lines of at most 200 characters. That section is terminated by end-of-file. Each line is a query. The query section will consist of no more than 200 lines.

For each query print the number of facts returned by the query as an integer. Values are to be separated from each other by newlines and/or spaces.

Problem 4
LogDB (continued)

Sample Input

```
test(arg1, arg2,arg3) test(1,2,3,not4)5(five) five_seven(john_smith,10_17_57)
foo(1,2,3,4) foo(5,6,7,8) foo(1,2,7,8)arc(80) foo(abc,xyz)
bar(1,2) bar (7,8) zoom(8,7)
arc(80) nofoo(alpha,d1,d2,d1,d4,d1)
foo(bar,spam) foo(more,less)

test(_,_,_,_)
arc(80)
  foo(bar,_)
  foo(_,spam)
  foo(_,less)
  nofoo(_,_p1,_,_p1,_,_p1)
  foo(bar,less)
foo(_,_,_,_)
foo(_,_,_30,_40)
foo(_,_,_30,_40)
foo(_,_,_30,_40)
foo(_x,_,_30,_40)
foo(_,_,_30,_40)
```

Output for the Sample Input

```
1
2
1
1
1
1
0
3
3
3
3
3
3
```

2020/2021 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 5
Digital Speedometer

A digital speedometer shows a vehicle's speed as integer miles per hour. There are occasions when the sensed speed varies between two integer values, such as during cruise control. Using a single threshold to round between adjacent integers often makes the display toggle rapidly between the two integers, which is distracting to the driver.

Your team must implement a smoothing technique for the display using separate rising and falling thresholds (t_r and t_f, respectively). See Figure 1 for a graphical depiction of the Sample Input for use with the following rules.

Each sensed speed, s, falls between two adjacent integers i and j, $i \leq s < j$, where $j = i + 1$. When displaying the sensed speed s as an integer:

- When s falls between i and $i + t_f$, s is displayed as i.
- When s falls between $i + t_r$ and j, s is displayed as j.
- When s falls between $i + t_f$ and $i + t_r$, s is displayed as i if the most recent preceding value for s outside of range $[i + t_f, i + t_r]$ is less than $i + t_r$, and s is displayed as j if the most recent preceding value for s outside of range $[i + t_f, i + t_r]$ is greater than $i + t_r$.
- Any sensed speed, $0 < s < 1$, must display as 1 because any non-zero speed, no matter how small, must display as non-zero to indicate that the vehicle is in motion.

The first line of input contains t_f, the falling threshold. The second line of input contains t_r, the rising threshold. The speed sensor reports s in increments of 0.1 mph. The thresholds are always set halfway between speed increments. All remaining lines until end-of-file are successive decimal speeds, s, in miles per hour, one speed per line. The third line of input, which is the first measured speed, will always be 0. There are at most 1,000 values of s in the input.

$$0 < t_f, t_r < 1; \quad t_f < t_r; \quad 0 \leq s \leq 120$$

Output is the list of speeds, one speed per line, smoothed to integer values appropriate to t_f and t_r.

Sample Input

```
0.25
0.75
0
2.0
5.7
5.8
5.7
5.2
5.7
0.8
0.2
```

Problem 5
Digital Speedometer (continued)

Output for the Sample Input

```
0
2
5
6
6
5
5
1
1
```

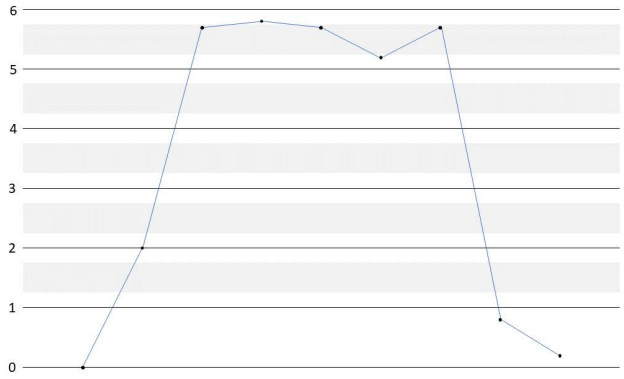


Figure 1. Sensor readings from the Sample Input, with $t_f = 0.25$ and $t_r = 0.75$.

Explanation of the Sample Data

Input	*Output*	*Explanation*
`0.25`		Value of t_f.
`0.75`		Value of t_r.
`0`	`0`	Initial input.
`2.0`	`2`	Input greater than 0, below threshold of 2.25.
`5.7`	`5`	Input greater than 2.0, in threshold range.
`5.8`	`6`	Input greater than 2.0, exceeds upper threshold of 5.75.
`5.7`	`6`	Input less than 5.8, in threshold range.
`5.2`	`5`	Input less than 5.8, below threshold of 5.25.
`5.7`	`5`	Input greater than 5.2, in threshold range.
`0.8`	`1`	Input greater than 0 and less than 1.
`0.2`	`1`	Input greater than 0 and less than 1.

2020/2021 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 6
Ada Loveslaces

A shoe has a lacing geometry: N, the number of eyelets (on a side), d, the distance between eyelets, s, the separation between the columns of eyelets when the shoe is laced, and t, the thickness of an eyelet. In Figure 1 below, $N = 3$, with eyelets numbered from 0 to $2N - 1$.

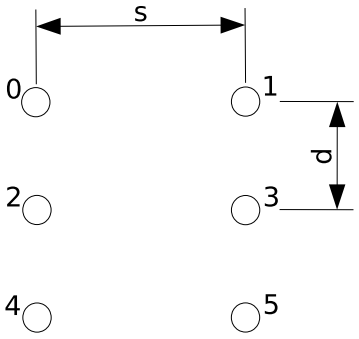


Figure 1. Eyelet numbering and dimensions for $N = 3$.

When laced, a shoelace of length L will have two free ends for knot tying. The free length of each end is f. The length of f must fall within a certain range, $[f_{min}, f_{max}]$ to accommodate a knot that is neither too small to tie nor so large as to dangle. See Figure 2 for a common criss-cross pattern.

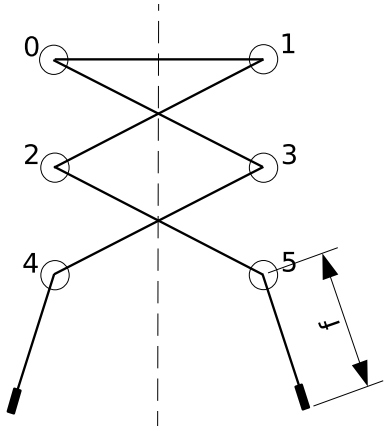


Figure 2. Criss-cross lacing pattern showing free end length f.

That special end of a lace that prevents fraying is called an "aglet." You're welcome.

Unfortunately, shoelaces break at the most inopportune times. Before purchasing a new shoelace, it is often necessary to retie the shorter, broken lace, or replace it with another lace borrowed from another shoe.

Problem 6
Ada Loveslaces (continued)

Given $N, d, s, t, f_{min}, f_{max}$, and a series of replacement shoelace lengths, L, for each shoelace length determine the number of lacing patterns that leave free ends, f, such that $f_{min} \leq f \leq f_{max}$. The rules of lacing are as follows:

- The resulting lacing pattern must be symmetric across a line drawn vertically between the even-numbered and odd-numbered eyelets.
- The lace can only pass through an eyelet at most once.
- The lace can only pass between eyelets in the same column that are immediately adjacent.
- The free ends of the lace must emerge from eyelet numbers $2N - 2$ and $2N - 1$.
- The lace must pass directly between eyelet numbers 0 and 1 to ensure the shoelace holds the shoe on the foot.

You are to assume that the surface the eyelets are on is to be treated as a plane, that the shoelace passes through the center of the eyelets, and that the thickness of the shoelace itself is negligible. The total length of the shoelace that is used for lacing is the length used between the eyelets plus t for each eyelet the shoelace passes through. In the example shown in Figure 2, $6t$ of the shoelace length is used by passing through the six eyelets.

The input begins with a single line containing $N, d, s, t, f_{min}, f_{max}$, separated by whitespace. Each additional line is a new value of L for which your program is to count the number of lacing patterns that meet the stated requirements. There will be between 1 and 100 values of L. All measurements are in integer millimeters.

N will be between 2 and 9 inclusive. d will be between 5 and 30 inclusive. s will be between 10 and 50 inclusive. t will be between 0 and 4 inclusive. f_{min} and f_{max} satisfy $0 \leq f_{min} \leq f_{max} \leq 2\,000$. Shoelace lengths will be between 200 and 2 000 millimeters inclusive. f_{min} and f_{max} will not be within a micron (0.001 millimeters) of the free end lengths of a valid lacing pattern.

For each shoelace length, your program is to print the number of possible lacing patterns that meet the above requirements. Values are to be separated from each other by newlines and/or spaces.

Sample Input

```
3 10 25 3 125 175
485
410
```

Output for the Sample Input

```
1
5
```

2020/2021 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 7
Codenames

Codenames is a popular board game. Two teams compete by each having a "spymaster" give one-word clues that can point to multiple words on the board. The other players on the team attempt to guess their team's words while avoiding the words of the other team. The objective is to be the first team to have all their team's words revealed.

Players split into two teams: red and blue. One player of each team is selected as the team's spymaster; the others are field operatives.

N Codename cards, each bearing a word, are laid out in a line. Each word represents one of the following: a red agent, a blue agent, an assassin, or an innocent bystander. All players can see all the Codename words, but only the spymasters know the identities of the cards.

Teams take turns. On each turn, the appropriate spymaster gives a verbal hint about the words on the respective cards. Each hint may only consist of one single word and a number. The spymaster's hint should be as close to their own agents' cards as possible. The hint's number tells the field operatives the maximum number of guesses to make.

After a spymaster gives the hint, their field operatives make guesses about which Codename cards bear words related to the hint and point them out, one at a time. When a Codename card is pointed out, the spymaster reveals the identity of that card—a blue agent card, a red agent card, an innocent bystander card, or an assassin card. Depending on the identity of the card, one of these things happens:

- If an assassin is pointed out, the game ends immediately, and their team loses.
- If an innocent bystander is pointed out, the turn simply ends.
- If an agent of the other team is pointed out, the turn ends, and that other team is one agent closer to winning.
- If an agent of their team is pointed out, they are one agent closer to winning, and they may choose to make another guess.

The game ends when all of one team's agents are identified (winning the game for that team), or when one team has identified an assassin (losing the game).

To simplify the problem, let's assume that both teams share the same dictionary of hint words and their associations with Codename cards. For example, consider this board ($N = 6$):

`Plate`(R) `Screen`(I) `Novel`(A) `Robin`(B) `Iron`(R) `Server`(B)
(R: Red team's agent; B: Blue team's agent; I: Innocent bystander; A: Assassin)

The list of hint words and their associated Codename cards are:

`Alfred`	`Robin, Server, Plate`
`Net`	`Server, Screen, Plate`
`Computer`	`Screen, Server`
`Twitter`	`Screen, Robin, Server`
`Crusoe`	`Robin, Novel`
`Film`	`Iron, Screen`

Problem 7
Codenames (continued)

Once the spymaster gives a hint word and a number, K (an integer between 1 and the number of unrevealed words associated with the hint), their field operatives make random guesses—with equal probability—from the list of associated words that are not revealed yet. They will continue to make guesses until one of the following happens: (1) they get K hits, (2) they have won the game, or (3) their turn ends with a miss. They will never make guesses outside the list or use hints from the previous rounds. It is illegal for the spymaster to give a hint word if all its associated Codename cards have already been revealed. All hint words can be used multiple times by either team.

For example, assuming the blue team goes first in the first round, the blue spymaster may give a hint "Twitter, 2". The blue team has 1/3 chance of guessing "Screen", "Robin" and "Server", respectively. If they happen to guess "Screen", their turn ends as the word is an innocent bystander. Otherwise, they get a hit and will make another guess, with 1/2 chance on either of the remaining two words. Regardless of the choice, their turn will end after this guess, and the red team will start their turn with the red spymaster giving a hint.

Now you are selected as the spymaster, and your team (color specified in the input) goes first. Assuming both spymasters play optimally, what is your probability of winning the game?

The first line of input consists of an integer and a character, separated by a single space. The integer is N ($1 \leq N \leq 15$), the size of the board. The character is `R` or `B`, indicating your team (red or blue).

The second line consists of N distinct words, separated by a single space. Each word consists of up to 20 lowercase English letters.

The third line consists of N single-character strings, separated by single spaces. The i-th string indicates the identity of the i-th word, with the following meaning: `R`—Red team's agent; `B`—Blue team's agent; `I`—Innocent bystander; `A`—Assassin. There are one or more red team agent and blue team agent words, and there are zero or more innocent bystander and assassin words.

The fourth line consists of an integer M ($1 \leq M \leq 50$), which is the number of hint words.

Each of the following M lines consists of an integer H_i ($1 \leq H_i \leq N$), followed by H_i distinct words separated by single spaces. Each line describes the Codename cards associated with the i-th hint word. All words are guaranteed to appear on the board.

Your program is to print the probability of winning the game. Your answer will be considered correct if it has an absolute error of at most 10^{-4} with the judges' data.

Sample Input

```
4 B
apple sleep java dog
B R I A
3
2 apple java
2 apple dog
2 sleep java
```

Output for the Sample Input

```
0.5000
```

Problem 7
Codenames (still continued)

Explanation for the Sample Data

The blue spymaster has three hint words to choose from (in any case, their team can only make one guess in the turn, so the choice of K does not matter):

1. If they choose the first, their team has 1/2 probability of guessing "apple" and winning the game, and 1/2 probability of guessing "java" and ending the turn, which allows the red spymaster to give the third hint word and win. So the blue team's winning chance is 1/2 in this case;
2. If they choose the second, their team has 1/2 probability of guessing "apple" and winning the game, and 1/2 probability of guessing "dog" and losing the game. So the blue team's winning chance is also 1/2;
3. If they choose the third, their team has 1/2 probability of guessing "sleep" and losing the game, and 1/2 probability of guessing "java" and ending the turn, which allows the red spymaster to give the third hint word and win. So the blue team's winning chance is 0 in this case.

To sum up, the best strategy for the blue spymaster is either (1) or (2) and their winning chance is 1/2.

2020/2021 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 8
Substring Characters

The set of distinct characters in a string is referred to as the generalized period of the string. As an example, the generalized period of the string "`aabbabb`" is {'`a`', '`b`'}.

A proper substring is a contiguous substring that is contained in a string and is not the string itself. So "`aabbabb`" is not a proper substring of the above example.

A minimal proper substring is one that can have no character removed from either end and still have the same generalized period. "`aabb`" is a proper substring of the example, but it is not minimal. "`ab`" is minimal.

Unique means that multiple occurrences of the same minimal proper substring in a string are only to be counted once. In the example, "`ab`" appears twice, but is counted once—hence the number of proper minimal unique substrings with the same generalized period of the entire string is two: "`ab`" and "`ba`".

Your team is to write a program to count the number of proper minimal unique substrings of a given string that have the same generalized period as the string itself. Input to your program is a series of lines terminated by end-of-file. Each line is a test case consisting of alphanumeric characters (a–z, A–Z, 0–9). Upper-case and lower-case letters are distinct. The new line character is not part of the test case string. No test case string will exceed 80 characters. There will be no more than 100 strings in the input.

For each input line print the number of proper minimal unique substrings of the input string. Values are to be separated from each other with newlines and/or spaces.

Sample Input

```
aabbabb
abAB34aB3ba7
104001144
aaabcaaa
a
bb
bd
1234567
```

Output for the Sample Input

```
2
1
3
2
0
1
0
0
```

2020/2021 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 9
Redundant Binary Notation

Redundant binary notation is similar to binary notation, except instead of allowing only 0s and 1s for each digit, we allow any integer digit in the range $[0, t]$, where t is some specified upper bound. For example, if $t = 2$, the digit 2 is permitted, and we may write the decimal number 4 as 100, 20, or 12. If $t = 1$, every number has precisely one representation, which is its typical binary representation. In general, if a number is written as $d_l d_{l-1} \ldots d_1 d_0$ in redundant binary notation, the equivalent decimal number is $d_l \cdot 2^l + d_{l-1} \cdot 2^{l-1} + \cdots + d_1 \cdot 2^1 + d_0 \cdot 2^0$.

Redundant binary notation can allow carryless arithmetic, and thus has applications in hardware design and even in the design of worst-case data structures. For example, consider insertion into a standard binomial heap. This operation takes $O(\log n)$ worst-case time but $O(1)$ amortized time. This is because the binary number representing the total number of elements in the heap can be incremented in $O(\log n)$ worst-case time and $O(1)$ amortized time. By using a redundant binary representation of the individual binomial trees in a binomial heap, it is possible to improve the worst-case insertion time of binomial heaps to $O(1)$.

However, none of that information is relevant to this question. In this question, your task is simple. Given a decimal number N and the digit upper bound t, you are to count the number of possible representations N has in redundant binary notation with each digit in range $[0, t]$ with no leading zeros.

Input consists of a single line with two decimal integers N ($0 \leq N \leq 10^{16}$) and t ($1 \leq t \leq 100$), separated by whitespace.

Output in decimal the number of representations the decimal number N has in redundant binary notation with each digit in range $[0, t]$ with no leading zeros. Since the number of representations may be very large, output the answer modulo the large prime 998 244 353.

Sample Input 1

```
4 2
```

Output for Sample Input 1

```
3
```

Sample Input 2

```
6 3
```

Output for Sample Input 2

```
4
```

Problem 9
Redundant Binary Notation (continued)

Sample Input 3

```
479 1
```

Output for Sample Input 3

```
1
```

Sample Input 4

```
3846927384799 62
```

Output for Sample Input 4

```
690163857
```

Sample Input 5

```
549755813887 2
```

Output for Sample Input 5

```
1
```

2020/2021 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 10
Staggering to the Finish

An oval track and field racing track consists of two parallel straightaway sections connected by two semicircles, depicted in Figure 1. Footraces run in the counterclockwise direction, ending at a common finish line located along the lower straightaway. For races that exceed the length of a single straightaway, starting lines must be staggered backwards, in the clockwise direction, from the finish line. The staggered starting lines must account for the curve of the semicircles and the widths of each running lane.

There are international standards for oval track dimensions. Unfortunately, the available area for a track doesn't always hold a standard track. Given the dimensions of the track and the length of the race, your team is to write a program to ensure equal race lengths by computing the staggered starting line positions.

The total distance of a race for any given lane is computed from the *line of running*. The line of running is an unmarked line to the right of the lane's inside marker (as seen from the counterclockwise direction). See Figure 2. For the innermost lane (lane 1) the line of running is usually farther from the lane marker than for the remaining lanes.

The track is mapped to an (x, y) coordinate system with $(0, 0)$ at the center of the track. See Figure 1.

The first line of input to your program contains seven values, $N\ R\ S\ W\ F\ L_1\ L_2$, separated by white-space, describing the geometry of a track, where:

- N is the integer number of lanes. $(1 \leq N \leq 9)$
- R is the inner radius of lane 1, a real value in meters. See Figure 1. $(1.0 \leq R \leq 100.0)$
- S is the length of the straightaway, a real value in meters. See Figure 1. $(1.0 \leq S \leq 200.0)$
- W is the width of each lane, a real value in meters. See Figure 2. $(0.5 \leq W \leq 3.0)$
- F is the x-coordinate of the finish line, measured from the centerline in Figure 1, a real value in meters. The finish line will always be in the lower (negative y) half of the track. $(|F| \leq S/2)$
- L_1 is the offset from the inner radius of lane 1 to the line of running for lane 1, a real value in meters. See Figure 2. $(0 \leq L_1 < W)$
- L_2 is the offset from the inner radius to the lines of running for lanes 2 and higher, a real value in meters. See Figure 2. $(0 \leq L_2 < W)$

The remaining lines until end-of-file specify D, the distance of a race, one race per line, a real value in meters. $(1.0 \leq D < 410.0)$ There will be at most 100 values of D.

Your program is to print a series of values for each race distance, separated from each other by spaces and/or newlines. Print the race distance first, followed by the (x, y) coordinates of the staggered starting line locations in lane number order. Express all values in meters. The (x, y) coordinate is the innermost point of a lane, NOT the line of running. Treat each lane *marker* (straightaway or radius) as a zero-width line. International standards require that the values be within 0.001 meters of the exact answer.

Problem 10
Staggering to the Finish (continued)

Sample Input

```
4 36.5 84.39 1.22 40.0 0.30 0.20
200.0
400
```

Output for the Sample Input

```
200.000 -40.0006 36.5000 -43.5119 37.6970 -47.3108 38.6025 -51.0664 39.1679
400.000 40.0012 -36.5000 46.9998 -37.4127 54.4292 -36.9682 61.4438 -35.2464
```

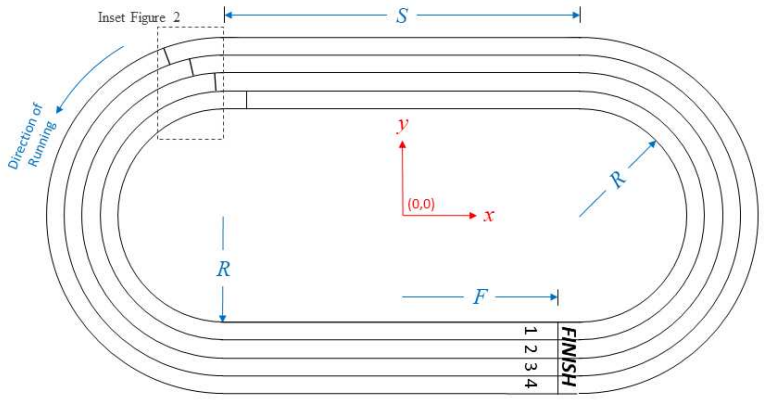


Figure 1. Oval track with 200m starting lines.

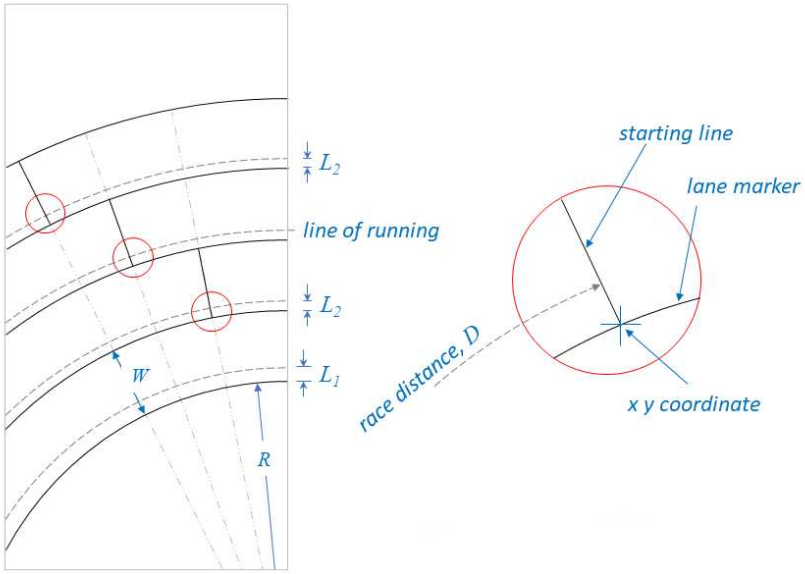


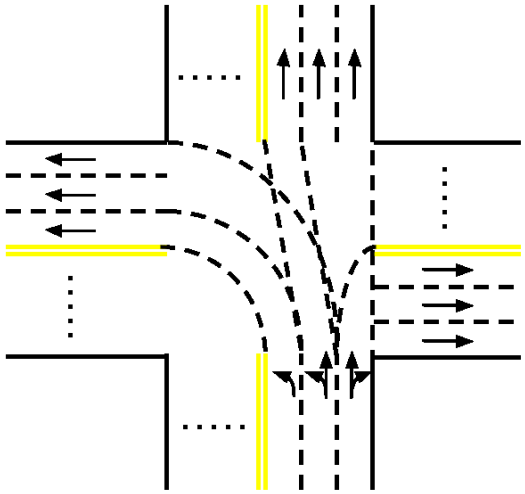
Figure 2. Inset showing staggered starting line locations.

2020/2021 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 11
Safest Taxi

Consider a town whose road network forms an $N \times M$ grid, where adjacent intersections are connected by roads. All roads are bi-directional. Each direction has an associated number—the time needed to travel from one end-point to another.

Each direction of each road consists of one or more lanes. A lane can serve one of the following functions: left-turn, straight, right-turn, or any combination of them. However, a left-turn lane cannot be placed to the right of a straight or right-turn lane, and a straight lane cannot be placed to the right of a right-turn lane. There are no U-turn lanes.



The rules for crossing intersections are illustrated in the above figure (suppose a car enters the intersection from the south). To make a left turn, it must be in one of the L left-turn lanes; let's number them 1 through L from left to right. The traffic rule says Lane i must turn into the i-th lane (counting from the left) of the target road, except that Lane L may turn into the L-th lane or any other lanes to its right.

Similarly, to go straight through an intersection, the car must be in one of the S straight lanes; let's number them 1 through S from left to right. Lane i must go into the i-th lane (counting from the left) of the target road, except that Lane S may go into the S-th lane or any other lanes to its right.

To make a right turn, the car must be in one of the R right-turn lanes. For the convenience of discussion, we consider these lanes and those of the target road *from right to left*. Let's number the right-turn lanes 1 through R from right to left. Lane i must turn into the i-th lane (counting from the right) of the target road, except that Lane R may turn into the R-th lane or any other lanes to its left.

It is guaranteed that if at least one left-turn / straight / right-turn lane is present, the target road must exist and have enough lanes to accommodate the left turn / straight / right turn, respectively. The time spent on crossing intersections is negligible.

In addition, a driver may change lanes in the middle of a road. Note that in the above rules for intersections, it doesn't count as a lane change to drive into any of the legal lanes of the target road. The time spent on lane changes is negligible.

A trip starts and ends at the rightmost lane of the midpoint of roads. The time needed to travel midpoint-to-endpoint is half of endpoint-to-endpoint.

Problem 11
Safest Taxi (continued)

You are running a taxi company called "Safest Taxi" in this town, with the slogan "your safety is in your hands". You let your customers choose the numbers X and Y for their trip, and the driver will make at most X left turns and Y lane changes to accomplish the trip.

What is the shortest time to fulfill each trip given the rules?

The first line of input consists of three integers N ($2 \leq N \leq 15$), M ($2 \leq M \leq 15$) and K ($1 \leq K \leq 3$), separated by a single space. The town's road network has N intersections north-south and M intersections west-east. Each road has K lanes.

The second line consists of a single integer D. The town's road network has D road segments. Every adjacent pair of intersections must appear in the list exactly once.

Each of the next D lines describes a road segment with the following format:

$$R_0\ C_0\ R_1\ C_1\ T\ L_0\ L_1 ... L_{K-1}$$

This describes a road segment going from the intersection at row R_0 column C_0 to the intersection at row R_1 column C_1 ($0 \leq R_0, R_1 < N$, $0 \leq C_0, C_1 < M$). Rows are numbered 0 through $N-1$ from north to south, and columns are numbered 0 through $M-1$ from west to east. The segment must connect two adjacent intersections, i.e., $\mid R_0 - R_1 \mid + \mid C_0 - C_1 \mid = 1$. The time to travel through the entire segment is T ($2 \leq T \leq 100$, T must be an even number). The next K strings describe the function of each of the K lanes, from left to right, with the following semantics:

`L`	Left-turn only
`S`	Straight only
`R`	Right-turn only
`LR`	Left-turn or right-turn
`LS`	Left-turn or straight
`SR`	Straight or right-turn
`LSR`	Left-turn, straight or right-turn

The next line consists of a single integer P ($1 \leq P \leq 50$), the number of trips to fulfill.

Each of the next P lines describes a trip with the following format:

$$R_{S0}\ C_{S0}\ R_{S1}\ C_{S1}\ R_{D0}\ C_{D0}\ R_{D1}\ C_{D1}\ X\ Y$$

This indicates that the starting point is the midpoint of segment $(R_{S0}, C_{S0}) \rightarrow (R_{S1}, C_{S1})$, and the destination is the midpoint of segment $(R_{D0}, C_{D0}) \rightarrow (R_{D1}, C_{D1})$. Both segments must appear in the above list. Both the starting point and the destination are on the rightmost lane. The customer requests that at most X ($0 \leq X \leq 4$) left turns and Y ($0 \leq Y \leq 4$) lane changes are allowed for the trip.

Output P lines. The i-th line contains a single integer which is the shortest time to fulfill each trip given the rules, or -1 if no feasible route exists.

Problem 11
Safest Taxi (still continued)

Sample Input

```
3 3 2
24
0 0 0 1 6 S R
0 1 0 0 8 L L
0 1 0 2 16 R R
0 2 0 1 18 LS S
0 0 1 0 8 LS S
1 0 0 0 8 R R
0 1 1 1 10 LS SR
1 1 0 1 16 L R
0 2 1 2 8 S R
1 2 0 2 8 L L
1 0 1 1 6 L SR
1 1 1 0 8 L R
1 1 1 2 16 L R
1 2 1 1 18 L SR
1 0 2 0 8 L L
2 0 1 0 8 S R
1 1 2 1 10 L R
2 1 1 1 8 LS SR
1 2 2 2 8 R R
2 2 1 2 8 LS S
2 0 2 1 10 LS S
2 1 2 0 12 R R
2 1 2 2 6 L L
2 2 2 1 8 S SR
6
2 1 1 1 1 1 1 0 1 1
2 1 1 1 1 1 1 0 1 0
2 1 1 1 1 1 1 0 0 0
0 1 0 2 0 2 0 1 2 0
1 0 0 0 0 0 1 0 2 0
2 1 2 0 2 0 2 1 2 0
```

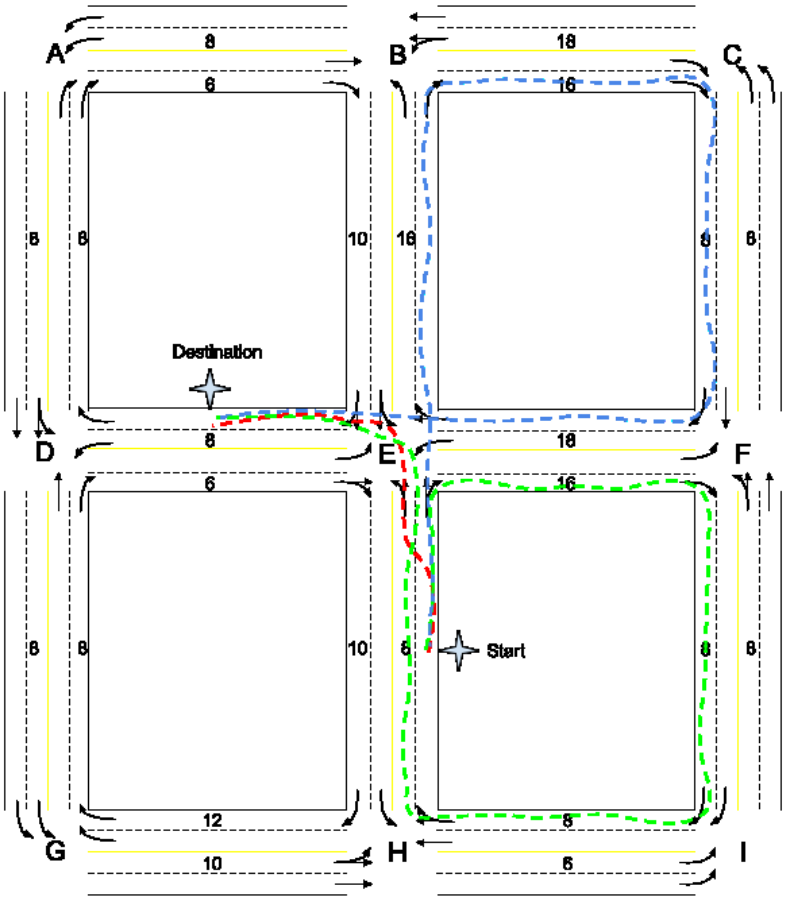
Output for the Sample Input

```
8
48
66
131
112
95
```

Problem 11
Safest Taxi (still continued)

Explanation for the Sample Data

The first three lines of the sample output are illustrated in the figure below.



- If $X = 1$ and $Y = 1$, the shortest path is shown in red: make a lane change before reaching E and make a left turn. The total time is $8/2 + 8/2 = 8$;
- If $X = 1$ and $Y = 0$, the shortest path is shown in green: go through E-F-I-H-E and make a left turn. The total time is $8/2 + 16 + 8 + 8 + 8 + 8/2 = 48$;
- If $X = 0$ and $Y = 0$, the shortest path is shown in blue: go through E-B-C-F-E. The total time is $8/2 + 16 + 16 + 8 + 18 + 8/2 = 66$.