2025/26 Problem Set

Problems

Do not open before the contest has started.

This page is intentionally left blank.

Problem A
A Little Leftover Pizza

The CS department has just had a big party and ordered too much pizza. Now it is time to put away the leftovers. They ordered a number of small, medium, and large pizzas, and there are still slices remaining in some or all of the pizza boxes. A small pizza comes in 6 slices, a medium pizza in 8 slices, and a large pizza in 12 slices. To save space, you can combine the leftover slices from the same size pizzas into a box of the right size, but you can't put a slice into a box for a different sized pizza, and you can't put more slices into a box than it originally held. What is the smallest number of boxes you will need to hold all the leftovers?

Input

The first line of input contains one positive integer n ($n < 1000$), the number of pizzas that were ordered. Each of the following n lines contains two items s_i and l_i (separated by a space) representing the leftovers for a given pizza. s_i is a string `S`, `M`, or `L` representing the size of pizza i, and l_i is an integer representing the number of leftover slices for pizza i. You can assume that each l_i is between zero and the original number of slices of that size pizza, inclusive.

Output

Output a single number, the fewest possible total boxes that can hold the leftover pizza according to the constraints given above.

Sample Input 1

```
3
S 0
M 5
L 0
```

Sample Output 1

```
1
```

Sample Input 2

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

Sample Output 2

```
2
```

Sample Input 3

```
4
S 1
M 1
M 3
L 1
```

Sample Output 3

```
3
```

Sample Input 4

```
4
L 6
M 2
M 6
L 6
```

Sample Output 4

```
2
```

Problem B
Andor Strikes Again

Cassian Andor is a rebel spy who has infiltrated the Empire's foremost starship armory. He has managed to hack into the facility's main computer where he's found various decision-making processes for the Empire's next big weapon—the Death and Boy Do We Mean Death Star (the name is a work in progress). Each decision-making process is in the form of an AND/OR tree, where leaves of the tree store boolean values and interior nodes are either AND nodes or OR nodes, alternating along any path from the root. An AND node evaluates to `true` if all its subtrees evaluate to `true` and evaluates to `false` otherwise; an OR node evaluates to `true` if at least one of its subtrees evaluates to `true` and evaluates to `false` otherwise. The value of a decision tree or any subtree is the value its root evaluates to. An example of an AND/OR tree (which evaluates to `true`) is shown in Figure B.1.

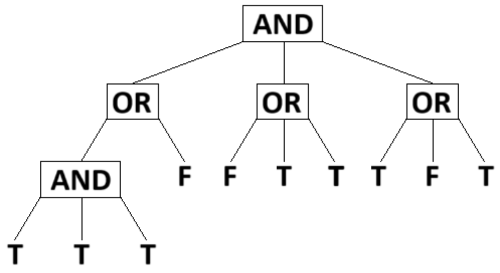


Figure B.1: Example AND/OR tree

Cassian has decided to sabotage each decision tree by making changes to one or more leaves in order to flip the value returned by the root. In order to make his subversion hard to detect he would like to change the minimum number of leaves. For example, in the tree above Cassian could change every leaf value to `false` to make the tree evaluate to `false`, but he can get the same result by changing just one of the left-most `true` leaves to `false`.

Though Cassian is a bit of a lone wolf, he could use a hand here. Help him out by writing a program that, given an AND/OR tree, determines the minimum number of leaves that need to be changed in order to change the evaluation of the tree.

Input

Input starts with a line containing two values n t, where n ($2 \leq n \leq 20$) is the number of levels in the tree and t is either `A` or `O` indicating the type of nodes in odd levels of the tree; nodes in even levels will be of the opposite type. The root of the tree is at level 1. Following this are n lines describing the AND/OR tree. Each line contains one or more entries $e_1\ e_2 \ldots e_m$, where each e_i is either the character `T` or `F`—indicating a leaf with value `true` or `false`—or a positive integer $v \leq 10$ indicating an internal node with v children. There is a single numeric value in the first of these lines; the number of entries in each subsequent line is equal to the sum of the numeric values in the previous level. Nodes in each level should be assigned left-to-right to nodes in the previous level. The total number of nodes in the tree, both internal nodes and leaves, is less than or equal to 10^5.

Output

Output the minimum number of leaves that need to be flipped to change the evaluation of the tree.

Sample Input 1

```
4 A
3
2 3 3
3 F T F T F T T
T T T
```

Sample Output 1

```
1
```

Sample Input 2

```
2 O
10
T T T T T T T T T T
```

Sample Output 2

```
10
```

Problem C
AROD

Since retiring from a lucrative athletic career, Alex has devoted most of his time to pondering foundational concepts in mathematics. Recently, he has been focusing on the categorization of triangles based on their interior angles, and has invented the acronym AROD to keep track of the four fundamental types:

- **A =** *acute:* all three angles are less than 90 degrees
- **R =** *right:* one angle is 90 degrees
- **O =** *obtuse:* one angle is greater than 90 degrees, but less than 180 degrees
- **D =** *degenerate:* one angle is 180 degrees, or, equivalently, the three vertices are collinear

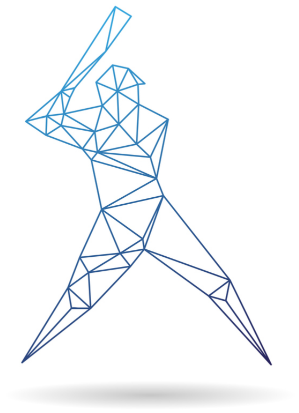


Image by Yakovliev (iStock); Used under license

Alex wonders how often three distinct vertices chosen from a regular grid of x-y points specify a triangle in each of the four AROD categories. More precisely, for positive integers m_x and m_y, he wants to consider all possible ways of choosing three distinct vertices from the set

$$V(m_x, m_y) = \{(x, y) : x \text{ and } y \text{ are integers, } 0 \leq x \leq m_x, 0 \leq y \leq m_y\}$$

and then categorize each of the corresponding triangles into one of the four categories listed above.

Input

The input is a line containing two positive integers, m_x and m_y, satisfying $m_x + m_y \leq 600$.

Output

Output four lines containing the numbers of times three distinct vertices chosen from $V(m_x, m_y)$ specify an acute, right, obtuse, or degenerate triangle, in that order (one number per line).

Sample Input 1	Sample Output 1

`1 2`	`0`
	`14`
	`4`
	`2`

Sample Input 2

```
2 3
```

Sample Output 2

```
22
94
84
20
```

Problem D
But I Want to Win

Borphee, the largest city in all of Frobozz, recently lost its mayor to the insatiable appetite of a Grue. The deputy mayor became the interim mayor and immediately took over the mayoral duties, which consist primarily of presiding over the annual *Double Fanucci Championships* (an extraordinarily complex card game) and the *From Bad to Worst Songfest*, which is where the most dreadful singers in the land gather every winter. The mayor's salary is several hundred thousand zorkmids (zm) a year, which is very lucrative for such an easy job. With the special election quickly approaching, it goes without saying that the deputy mayor wants to do anything in his power to change his "interim mayor" status to "mayor" and retain his very generous salary.

The entire empire uses a variation of *single-winner ranked choice voting* for all its elections. It works like this:

1. Voters mark their ballots in order of preference: first choice (their favorite candidate), second choice (their next favorite candidate) and so on. Each voter **must** rank all candidates on their ballot. For example, if there are 12 candidates in the election, then each voter will rank their choices from 1 (their favorite) to 12 (their least favorite). The election then proceeds in a set of *rounds*.
2. At the start of each round, all first-choice votes are counted. If a candidate receives more than 50% of the first-choice votes, that candidate wins, and the election is over.
3. If there is no winner, then the candidate with the least number of first-place votes is removed from all ballots, and the remaining candidates with worse rankings move up. For example, if a voter ranked four candidates in the order A (first-choice) B C D and candidate A is removed, then this voter's ballot is changed to B (new first-choice) C D; if candidate B is removed, then the ballot is changed to A C D. If one or more candidates tie for the least number of first-place votes then all of them are removed from the ballots.
4. After the ballots are readjusted, return to step 2 to start the next round.

This procedure may continue until there are only 2 candidates left, at which point the candidate with the majority of first-place votes wins.

The interim mayor assumes that the election will eventually come down to him and one other very popular candidate as the two front runners. With this in mind, he wants to concentrate his campaigning efforts on the voters who ranked one of the other candidates first on their ballot, in case the initial vote count puts him in 2nd place. To help him better plan, he wants you to develop a program that will allow him to determine the minimum number of rounds (after the first round) that it will take for him to obtain a majority of the votes, assuming he is in 2nd place after round 1.

Input

Input starts with a line containing an integer, c ($2 \le c \le 20$), which is the number of candidates in the election. This is followed by a single line containing c distinct integers $v1, v2, \ldots, vc$, where each vi ($1 \le vi \le 10^9$) specifies the number of votes candidate i received.

Output

Output consists of a single line. If the second place candidate can not win the election, then output `IMPOSSIBLE TO WIN`. Otherwise, output the minimum number of rounds (after the first round) that it will take for the second place candidate to win the election using ranked choice voting.

Sample Input 1

```
3
1 70 67
```

Sample Output 1

```
IMPOSSIBLE TO WIN
```

Sample Input 2

```
6
1 2 13 12 70 69
```

Sample Output 2

```
3
```

Sample Input 3

```
3
11 2 10
```

Sample Output 3

```
1
```

Sample Input 4

```
3
1 2 3
```

Sample Output 4

```
IMPOSSIBLE TO WIN
```

Problem E
Chess Solitaire

Chess Solitaire is a version of chess meant to be played by a single person. Okay, you probably figured that out on your own, but the exact rules are as follows: given a board with 2 or more chess pieces, none of which are pawns, find a series of captures which result in a single piece left on the board. Any piece that is moved must capture another piece, so if there are initially m pieces on the board, the solution involves $m - 1$ moves. An example puzzle and its solution is shown in Figure E.1 (which corresponds to Sample Input 1):

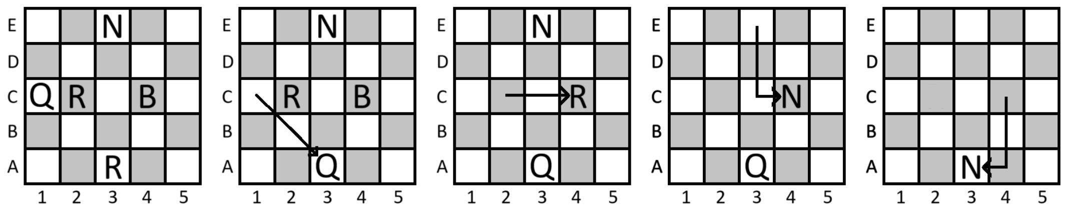
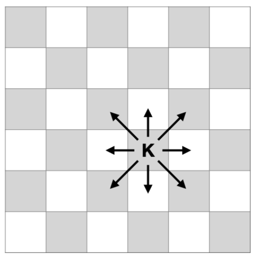
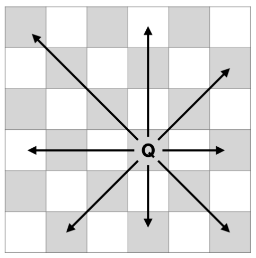


Figure E.1: Chess solitaire problem and a solution.

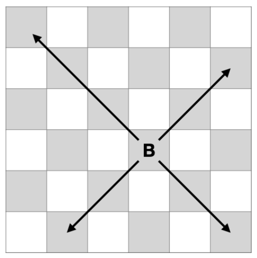
As a reminder, here are how the various chess pieces can capture other pieces:



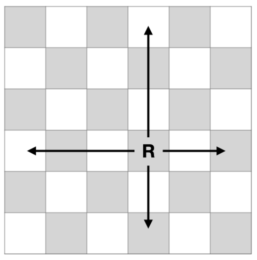
A king may move one square in any of the eight cardinal and diagonal directions.



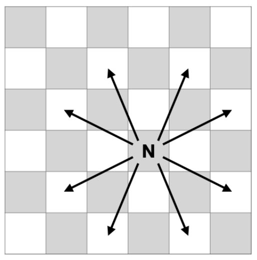
A queen may move any number of squares in any of the eight cardinal and diagonal directions, but may not jump over other pieces.



A bishop may move any number of squares in any of the four diagonal directions, but may not jump over other pieces.



A rook may move any number of squares in any of the four cardinal directions, but may not jump over other pieces.



A knight may move exactly two squares in one cardinal direction plus one square in a perpendicular direction, and may jump over other pieces.

For this problem, you will be given a chess solitaire puzzle and output a series to solve that puzzle.

Input

The input begins with a line n m where n ($2 \le n \le 8$) is the number of rows and columns in the board, and m ($2 \le m \le 10$) is the number of pieces in the problem. Following this are m lines of the form p loc, where p is one of `'N'`, `'B'`, `'R'`, `'Q'`, `'K'` indicating knight, bishop, rook, queen and king, respectively, and loc is a string of length 2 indicating the location of the piece on the board. The first character is an upper case letter indicating the row and the second character is an integer indicating the column (as shown in the figure above). All locations are valid and no two locations will be the same.

Output

Output $m-1$ lines displaying the $m-1$ moves to solve the problem. Each line will be of the form p loc_1 `->` loc_2 indicating a move of piece p from location loc_1 to location loc_2. If there are multiple solutions, print the one which has a first move with the lexicographic lowest loc_1. If there is still a tie, print the one which has a first move with the lexicographic lowest loc_2. If there is still a tie, apply these rules to the second move, and so on.

If there is no possible solution to the puzzle, output `No solution`

Sample Input 1

```
5 5
N E3
Q C1
R C2
B C4
R A3
```

Sample Output 1

```
Q: C1 -> A3
R: C2 -> C4
N: E3 -> C4
N: C4 -> A3
```

Sample Input 2

```
4 4
Q D4
Q B1
Q C2
K A3
```

Sample Output 2

```
No solution
```

Problem F
Fractional Sequence

Consider the following increasing sequence, S, of rational numbers:

$$1,\ 2,\ 2\frac{1}{2},\ 3,\ 3\frac{1}{3},\ 3\frac{2}{3},\ 4,\ 4\frac{1}{4},\ 4\frac{1}{2},\ 4\frac{3}{4},\ 5,\ 5\frac{1}{5},\ 5\frac{2}{5},\ 5\frac{3}{5},\ 5\frac{4}{5},\ 6,\ \ldots.$$

S is composed of an infinite set of blocks, $N_1, N_2, N_3, \ldots$, where block N_i is

$$i,\ \ i+1/i,\ \ i+2/i,\ \ \ldots,\ \ i+(i-1)/i.$$

So $S(1) = 1, S(2) = 2, S(3) = 2\frac{1}{2}$, etc. Write a program which takes as input an integer n and outputs $S(n)$.

Input

Input is a single line containing an integer, n ($1 \le n \le 4 \cdot 10^9$).

Output

Output $S(n)$ as a single integer if the answer is a whole number. Otherwise, output the integer part, a single space and a proper fraction a/b in lowest terms (i.e. $0 < a < b$ and $GCD(a, b) = 1$). See the sample outputs.

Sample Input 1	Sample Output 1
`326`	`26`

Sample Input 2	Sample Output 2
`448`	`30 2/5`

Sample Input 3	Sample Output 3
`4000000000`	`89443 19596/89443`

Problem G
How Many Balls?

If a bag contains r red balls and g green balls and two balls are drawn at random, the probability of getting one ball of each color is

$$P(r,g) = \frac{2rg}{(r+g)(r+g-1)}$$

Write a program which takes as input a rational number p/q in lowest terms and determines whether there is a number $r \leq 10^6$ and a $g \geq r$ for which $P(r,g) = p/q$.

Input

The only line of input contains two space-separated positive integers p ($p > 0$) and q ($2p - 1 \leq q \leq 1000$). These two integers are guaranteed to be relatively prime.

Output

If there is a solution, print the two positive integers r and g satisfying the conditions above, separated by a space. If there are multiple solutions, output the one with the smallest r value. For any r value, there is at most one g value ($g \geq r$), which solves $P(r,g) = p/q$. If there is no solution with $r \leq 10^6$, print the word `impossible`.

Sample Input 1	Sample Output 1
`12 25`	`9 16`

Sample Input 2	Sample Output 2
`8 25`	`impossible`

This page is intentionally left blank.

Problem H
Move It, Slowpoke!

Centerville has a bit of a problem. Because of recent road construction many slow-moving vehicles (garbage trucks, delivery trucks, etc) have been rerouted through town. Many of the citizens (or at least this problem contributor) are becoming increasingly upset about being stuck behind these vehicles, especially when they stay on one continuous road for an extended period of time. The town council has recently enacted an ordinance stating that no slow-moving vehicle can stay on any two or more roads considered "continuous" past a distance d. The details of the ordinance are the following:

1. The ordinance begins with a definition of what a "slow-moving vehicle" is. That's not important for us here (though you know one when you see one).
2. The ordinance next lists the ordered pairs of roads in town that are considered "continuous". Slow-moving vehicles are allowed to drive on either road in a pair separately (even if the length of either road is $> d$), but cannot drive on the first road followed immediately by the second road if the total length of the pair is $> d$. Note that if the second road of one continuous pair matches the first road of another continuous pair, then the three roads in the two pairs are together considered continuous. In other words, if (say) $\texttt{A} \to \texttt{B} \to \texttt{C}$ is considered continuous and $\texttt{B} \to \texttt{C} \to \texttt{D}$ is considered continuous, then $\texttt{A} \to \texttt{B} \to \texttt{C} \to \texttt{D}$ is considered continuous.
3. The ordinance ends with methods to determine d (again, these details are not important to us here).

Obviously the owners of these slow-moving vehicles are a bit perturbed (serves them right I say, but again that's not important to us). While finding the shortest path between two points used to be straight-forward (no pun intended), now it's become a bit of a challenge given the restrictions placed on them—in fact, it may not even be possible to get from one point to another. For example, consider the set of roads in Figure H.1. Assume that a slow-moving truck must get from intersection A to intersection D, and that three pairs of roads are considered continuous: $\texttt{A} \to \texttt{B} \to \texttt{C}$; $\texttt{A} \to \texttt{B} \to \texttt{E}$; and $\texttt{B} \to \texttt{F} \to \texttt{G}$. If d is 30 (or higher), then the truck can drive $\texttt{A} \to \texttt{B} \to \texttt{C} \to \texttt{D}$. If d is 25, this route is no longer usable, so the shortest path is now $\texttt{A} \to \texttt{B} \to \texttt{E} \to \texttt{C} \to \texttt{D}$ (Sample Input 1). If d is 15, then $\texttt{A} \to \texttt{B} \to \texttt{E}$ is not longer usable, so the shortest route is now $\texttt{A} \to \texttt{B} \to \texttt{F} \to \texttt{G} \to \texttt{C} \to \texttt{D}$ (note that in this case you are allowed to drive from A to B even though the length of that road is $> d$). Finally, if d is < 14, then it is impossible for this truck to drive from A to D (Sample Input 2). Note that slow-moving vehicles are unable to perform u-turns without causing even more delays, so, for example, the path $\texttt{A} \to \texttt{B} \to \texttt{F} \to \texttt{B} \to \texttt{C} \to \texttt{D}$ is not possible for any value of d.

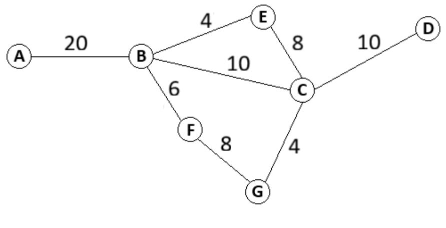


Figure H.1: Example road network, used by Sample Inputs 1 and 2.

Given a specification of the road network, two intersections s and t, and a value for d, the owners of the slow-moving vehicles would like to know the shortest distance to travel from intersection s to intersection t.

Input

Input starts with a line containing 6 integers n m k d s t, where n ($2 \leq n \leq 100$) is the number of intersections in the town (numbered 1 to n), m is the number of roads between intersections, k ($0 \leq k \leq m(m-1)$) is the number of ordered pairs of roads that are considered as continuous when driven consecutively, d ($1 \leq d \leq 100$) is the longest distance that slow-moving vehicles can stay on a continuous sequence of roads, s ($1 \leq s \leq n$) is the intersection to travel from, and t ($1 \leq t \leq n, s \neq t$) is the intersection to travel to. Following this are m lines, consisting of three integers a, b, ℓ ($1 \leq a, b \leq n$, $a \neq b$, $1 \leq \ell \leq 100$) indicating a two-way road of length ℓ connects intersections a and b. There is at most one road between any two intersections. Following this are k lines, of the form a b c ($a \neq b, a \neq c, b \neq c$) indicating that driving on the road from a to b followed by the road from b to c is considered as continuous driving. It is guaranteed that there is a road between a and b and that there is a road between b and c. Note that this does not mean that the reverse order of driving on those roads is considered continuous driving.

Output

If it is impossible to travel from s to t output `impossible`; otherwise output the shortest distance to travel from s to t.

Sample Input 1

```
7 8 3 25 1 7
1 2 20
2 3 10
2 4 4
4 3 8
2 5 6
5 6 8
6 3 4
3 7 10
1 2 3
1 2 4
2 5 6
```

Sample Output 1

```
42
```

Sample Input 2

```
7 8 3 12 1 7
1 2 20
2 3 10
2 4 4
4 3 8
2 5 6
5 6 8
6 3 4
3 7 10
1 2 3
1 2 4
2 5 6
```

Sample Output 2

```
impossible
```

Problem I
Number Pyramid

Uneedtoth, Inc. is a company that makes mathematical games for elementary school students, and Pam D. Monium has just been put in charge of their latest product: Number Pyramids. The game consists of cards each containing a pyramid of squares, some filled with numbers and some empty (see the first card on the left in Fig I.1). The object of the game is to find values that go in each of the empty squares so that the value in every square (except those on the bottom) is the sum of the values in the two squares below it. To keep things, well, elementary, the final values in the squares are always integers that are < 100 in absolute value. The cards are also set up so that they can be solved by multiple passes through the card, each pass going from the top to the bottom row and left to right in each row, filling in any value that can be determined by its neighbors. An example of this is shown in the remaining three cards in Fig I.1. Note that there may be other ways to solve a given card, but if a unique solution cannot be found using this method, the card should not be issued to the students. Pam calls cards like this "ambiguous".

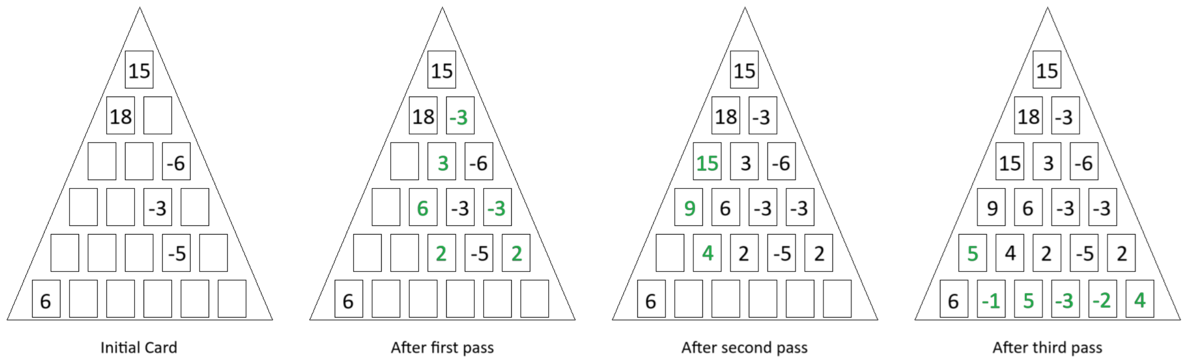


Figure I.1: Sample Number Pyramid card and one way to solve it.

Things seemed to be going smoothly for Pam, with machines in Uneedtoth's factory spitting out card after card. But when Pam started doing some product testing she noticed that there was a glitch in the machine and some of the cards were either ambiguous, or led to contradictory values for one or more squares. For example, if the card on the left in Fig I.1 was missing the 6 in the lower left corner, that card would be ambiguous as it would allow multiple solutions; if that card had a 5 in the rightmost square in the last row, or a 3 in the rightmost square in the second from the top row, each of these would lead to a contradiction and the card would have no solution. Finally, if the 18 was changed to a -90, there would be no solution as one of the values solved for would be outside of the range $-99, \ldots, 99$

As Pam sees it, there are two paths she could take now. The first is to ignore the testing and ship out the cards, relying on American children's terrible math skills to give her enough time to find another job; or she can have someone write software to detect the bad cards and remove them. After much soul searching and a look at her bank account she's decided on the latter and has come to you for help.

Input

The input begins with a line n ($2 \leq n \leq 100$) which is the number of rows on a pyramid card. Following this are n lines specifying the rows of the pyramid. The first line containing a single value, the second line two values, and so on. All values v will be integers in the range $-99 \leq v \leq 100$, where the special value of 100 indicates an empty square.

Output

Output one of three things: if the card has multiple solutions with all values in the range $-99, \ldots, 99$, output `ambiguous`; if the card has no solution, or has a solution with one or more values outside the range $-99, \ldots, 99$, output `no solution`; otherwise output `solvable` and then output the solution using the format shown in the Sample Output.

Sample Input 1

```
6
15
18 100
100 100 -6
100 100 -3 100
100 100 100 -5 100
6 100 100 100 100 100
```

Sample Output 1

```
solvable
15
18 -3
15 3 -6
9 6 -3 -3
5 4 2 -5 2
6 -1 5 -3 -2 4
```

Sample Input 2

```
2
100
10 100
```

Sample Output 2

```
ambiguous
```

Sample Input 3

```
2
10
10 10
```

Sample Output 3

```
no solution
```

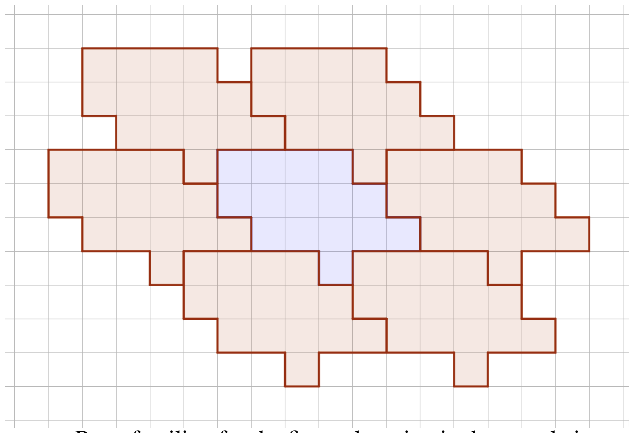
Sample Input 4

```
2
100
51 51
```

Sample Output 4

```
no solution
```

Problem J
Polyomino Tiling


Part of a tiling for the first polyomino in the sample input.

A polyomino is a nonempty connected union of unit squares without holes, where the corners of each unit square are at integer coordinates in the two-dimensional plane. A polyomino can be described by a string representation of its boundary. Starting at some integer coordinates on the polyomino's boundary we can follow the boundary by taking unit steps up, right, down, or left. We represent each of these steps with the letters `u`, `r`, `d`, `l`, respectively. If we concatenate these letters while following the boundary of the polyomino until we reach the starting coordinates, then the concatenation is called the boundary string for the polyomino. Note that while following the boundary, only the starting coordinates may be visited twice, all other coordinates in the path must be visited exactly once.

The cyclic rotations of the boundary string are obtained by starting at different coordinates on the boundary and proceeding in the same order as in the boundary string (either clockwise or counter-clockwise, but not both). For example, the cyclic rotations of `urdl` are `urdl`, `rdlu`, `dlur`, and `lurd`.

For a string S, consisting of letters `u`, `r`, `d`, and `l`, define $\overline{S}$ as the string obtained by reversing the letters of S and changing each `u` to `d`, each `d` to `u`, each `r` to `l`, and each `l` to `r`. For example, for $S = \texttt{uruurrdl}$, we have $\overline{S} = \texttt{rulldddld}$.

It can be proven that a polyomino will tile the plane by translations (i.e., without rotations nor reflections) if and only if a cyclic rotation of the boundary string B can be written as $B = X \cdot Y \cdot Z \cdot \overline{X} \cdot \overline{Y} \cdot \overline{Z}$ or $B = X \cdot Y \cdot \overline{X} \cdot \overline{Y}$, where X, Y, Z are nonempty strings. Quite often, there may be many ways to rotate the boundary string and write it in one or both of these two forms.

Given a boundary string for a polyomino, count the number of ways that each cyclic rotation B of the boundary string can be written as either $B = X \cdot Y \cdot \overline{X} \cdot \overline{Y}$ or $B = X \cdot Y \cdot Z \cdot \overline{X} \cdot \overline{Y} \cdot \overline{Z}$ where X, Y, and Z are nonempty strings. This count will be 0 if the polyomino cannot tile the plane by translations.

Input

Input consists of a single line containing two values k s, where k ($4 \leq k \leq 10\,000$) is the length of the boundary string and s is the boundary string. The boundary string will consist of the letters `u`, `r`, `d`, and `l`.

Output

Print a single integer, the number of ways that each cyclic rotation of the boundary string can be written in the form $X \cdot Y \cdot \overline{X} \cdot \overline{Y}$ or $X \cdot Y \cdot Z \cdot \overline{X} \cdot \overline{Y} \cdot \overline{Z}$.

Sample Input 1	Sample Output 1
`20 uurrrrdrdrdlldlullul`	`6`

Sample Input 2	Sample Output 2
`14 ururdrrdldllul`	`4`

Sample Input 3	Sample Output 3
`12 uurdrurddlll`	`0`

Sample Input 4	Sample Output 4
`8 urrrdlll`	`16`

Problem K
Triptych

Image from the National Gallery of Art; Public Domain

Donald is the curator of a museum specializing in medieval artifacts. He is especially proud of a recently acquired *triptych* (a three-paneled work of art) that is filled with religious iconography. He wants to make this triptych the main attraction in the museum's next exhibit, and he plans to highlight one of the three panels during each week of the exhibit by shining a special spotlight on it.

As an intern in the museum's IT department, your primary task is to program the microcontroller that operates the spotlight. For simplicity, the three panels are labelled A, B, C, from left to right, and the input to the microcontroller will be a string of these characters, specifying, in order, the panel that is to be highlighted during each week of the exhibit (which may run as long as a year). Such a string is called a *spotlight sequence.*

In addition to programming the microcontroller, Donald has asked you to generate some possible spotlight sequences, from which he will pick one. However, not every sequence of characters from $\{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$ is acceptable, since Donald is very strict about the following rules:

1. *variety* — The same panel cannot be highlighted during two consecutive weeks, except for the middle panel (B), which is designed to be (literally) the center of attention; the middle panel can be highlighted during two consecutive weeks, *but not during three or more consecutive weeks.*
2. *balance* — No panel should be highlighted significantly more often than any other panel. More precisely, Donald has chosen a non-negative integer, d, and you must ensure that any two of a, b, c differ by at most d, where a, b, and c are the numbers of occurrences of A, B, and C, respectively, in any given spotlight sequence.
3. *aibohphobia* (fear of palindromes) — Donald has an extreme aversion to palindromes (no one dares ask why), so no spotlight sequence can be a palindrome (the same forward and backward).

Given the number of weeks of the exhibit, and the value of d, determine the total number of spotlight sequences you can generate for Donald to choose from.

Input

The input consists of a line containing two integers, w $(2 \leq w \leq 52)$ and d $(0 \leq d \leq 10)$, the number of weeks of the exhibit and the "maximum difference" value described above, respectively.

Output

Output a single integer: the total number of spotlight sequences of length w you can generate that comply with all of Donald's rules.

Sample Input 1

```
3 0
```

Sample Output 1

```
6
```

Sample Input 2

```
2 0
```

Sample Output 2

```
0
```

Sample Input 3

```
4 1
```

Sample Output 3

```
24
```

Problem L
Valley Gulls

The Cliff Valley complex is so called because of the steep, vertical cliffs that enclose its series of valleys and hills. Besides this unique geographical layout, one of the things it is most known for are the Blue-Throated Mountain Gulls which make their nests on the scenic hills in the complex. The other thing it is known for is rain — lots of rain. This can make it tough on the Blue-Throated Mountain Gulls as often their nests have to be abandoned due to rising flood waters. Famous for their unique beauty and haunting bird song, the species is not the brightest when it comes to choosing nesting habitats.

Environmentalist Audrey Bond has been put in charge of modeling the flooding in the valley and the threat this poses to the gulls. She has decided to go with simple 2-D slices of the valley to make things simpler. She has somewhat limited information on the layout of the valley and only has a small set of elevation points from a previous survey with which to work, the first and last of which are located at the cliff walls (see Figure L.1a). She also knows that no nests are at these points, as the Blue-Throated Mountain Gull can get pretty unruly when a surveyor gets close to their nest. Given these points, Audrey intends to fit a piece-wise, first-degree interpolation curve to model each valley slice. In other words, she's going to draw straight lines between each pair of consecutive points. An example is shown in Figure L.1b.

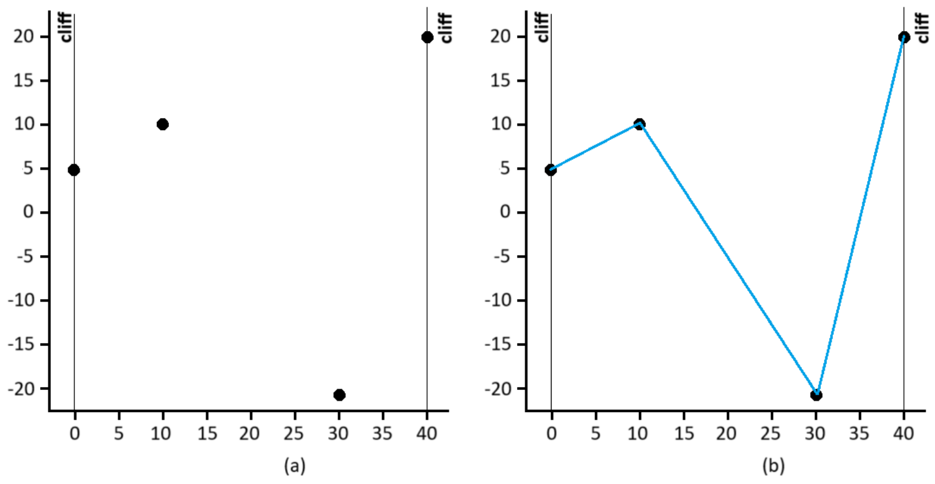


Figure L.1: (a) Survey points (Sample Input 1). (b) Straight line connection of points

.

For a given rate of rainfall, Audrey figures it will be easy to determine how long it will take flood waters to reach any given Blue-Throated Mountain Gull nest in the valley. Or at least she thought it would be easy — turns out even modeling with straight lines is a little harder than she thought. Can you help a gull out?

Input

The first line of input contains three values p r m where p $(2 \leq p \leq 50)$ is a positive integer indicating the number of survey points, r $(0.0 < r \leq 3.0)$ is the rate of rainfall in feet per hour, and m $(1 \leq m \leq 100)$ is a positive integer indicating the number of nests. The value of r will have at most two digits after the decimal point. Following this is a single line with p pairs of integer values x_i y_i $(0 \leq x_i \leq 5000, -1000 \leq y_i \leq 1000)$ specifying the p survey points, in feet. No two survey points will have the same x or y value. The x values are in ascending order and no three consecutive points will be collinear. The first and last x

values correspond to the locations of the cliff and you may assume that the heights of the cliff sides are greater than all the y values of the survey points. The final line of input contains m integers $n_1\ n_2 \ldots n_m$ $(x_1 < n_1 < n_2 < \cdots < n_m < x_p)$ indicating the x values of m nest locations. None of the corresponding y values for these x values equal any y value of a survey point.

Output

Output m lines, one for each of the corresponding m nest locations, specifying the time in hours that flood waters would reach the nest, assuming the rain starts at time $t = 0$ and continues at a steady rate of r. Answers will be judged correct if they have an absolute or relative error no worse than 10^{-5}.

Sample Input 1	**Sample Output 1**

`4 0.2 2`
`0 5 10 10 30 -20 40 20`
`6 18`
	`4.50000000`
	`21.68750000`