Problem A
Nine Knights

In the game of chess, knights are unique due to their "L-shaped" movement. A knight can move, as shown in Figure A.1, by either moving two squares sideways and one square up or down, or moving one square sideways and two squares either up or down.

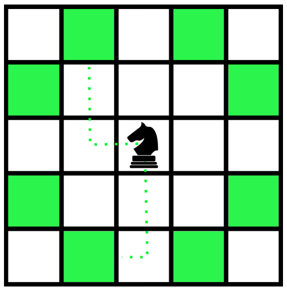


Figure A.1: The highlighted squares show all possible moves for a knight.

In the Nine Knights puzzle, exactly nine knights must be positioned on a 5-by-5 board so that no knight can attack another knight with a single move. The configuration shown in Figure A.2 is an invalid solution because two of the knights can attack each other, where the configuration shown in Figure A.3 is a valid solution.

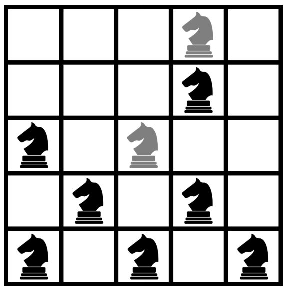


Figure A.2: Invalid game configuration

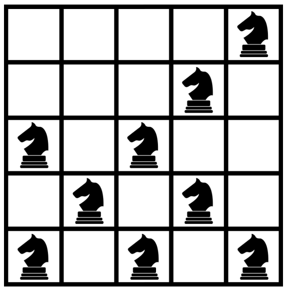


Figure A.3: Valid game configuration

Given the description of a game configuration, your job is to determine whether or not it represents a valid solution to the Nine Knights puzzle.

Input

The input will consist of 5 lines, each having 5 characters. All characters will be either 'k', indicating the placement of a knight, or '.', indicating an empty space on the board.

Output

Display the word `valid` if the given chess board is a valid solution to the Nine Knights puzzle. Otherwise, display the word `invalid`.

Sample Input 1

```
...k.
...k.
k.k..
.k.k.
k.k.k
```

Sample Output 1

```
invalid
```

Sample Input 2

```
.....
...k.
k.k.k
.k.k.
k.k.k
```

Sample Output 2

```
valid
```

Sample Input 3

```
.....
...k.
k.k.k
.k.k.
k...k
```

Sample Output 3

```
invalid
```

Problem B
Batter Up

While the Chicago Cubs were ecstatic with their 2016 World Series championship, they were eliminated from the playoffs in 2017. Looking ahead to 2018 they are beginning to embrace the more data-driven analysis of player's values known as Sabermetrics.

For example, a player's batting average is calculated by dividing the total number of base hits by the total number of official at-bats. One limitation of using the batting average to evaluate players is that it treats all hits equally, rather than taking into account doubles, triples or home runs. For this reason, analysts often prefer to consider what is known as the *slugging percentage*, which distinguishes between different hit outcomes. To calculate the slugging percentage, the total number of *bases* of all hits is divided by the total numbers of time at bat, that did not result in walks, or *at-bats*.

More specifically, an at-bat can earn 0, 1, 2, 3 or 4 bases (these are referred to as *official* at-bats). Furthermore, some at-bats, such as those that result in a base-on-balls (i.e., a "walk") are not considered in either the player's batting average or slugging percentage.

For example, if a player hits a triple (3 bases), strikes out (0 bases), and hits a double (2 bases), their slugging percentage would be $\frac{3+0+2}{3} \approx 1.6667$. If a player hits a single (1 base), walks, and hits a home run (4 bases), the slugging level would be $\frac{1+4}{2} = 2.5$. Notice that in this case, the denominator is two, not three, because the walk does not count towards the slugging percentage.

Input

The input is composed of two lines. The first line contains a single positive integer n ($1 \leq n \leq 100$) that specifies the number of at-bats. The second line contains n integers, separated by spaces, each describing one of those at-bats. Strike-outs, singles, doubles, triples, and home runs are represented as `0`, `1`, `2`, `3`, `4`, respectively. Walks are represented as `-1`. You may assume that there will always be at least one official at-bat (i.e., at least one at-bat will *not* be a walk).

Output

Display the player's slugging percentage as a real number, accurate to within an absolute or relative error of 10^{-3}. We recommend that you do *not* round the value that you calculate.

Sample Input 1	Sample Output 1

`3`
`3 0 2`
	`1.6666666666666667`

Sample Input 2	Sample Output 2

`3`
`1 -1 4`
	`2.5`

Sample Input 3	Sample Output 3

`11`
`-1 -1 -1 -1 0 0 0 0 0 0 1`
	`0.14285714285714285`

Problem C
Ring String

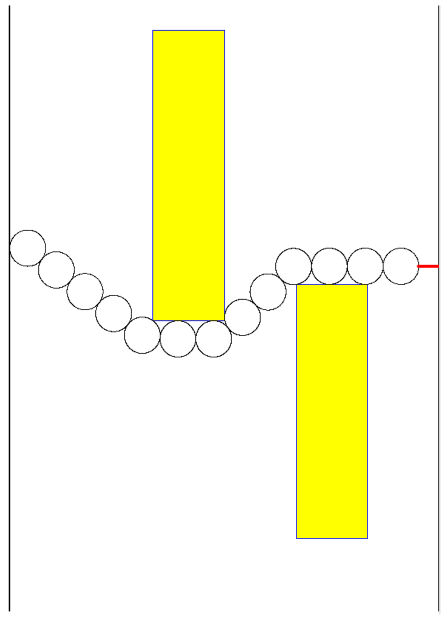


Figure C.1: A strand of disks along a wall

Suppose you want to decorate a party room wall with a string of rings, each having 1-foot *diameter* and each touching the next ring at precisely one point. You cannot necessarily string the rings in a straight line because there are rectangular pictures on the wall that you must avoid. From a given starting point against the left end of the wall you want to use as few rings as possible so that the last ring is less than one foot from the right end of the wall. Using the minimum number of rings, you also wish to minimize the distance between the final ring and the right wall.

Figure C.1 shows such an optimal solution, with the minimum number of rings and with the distance to the right wall (shown as a red segment) minimal among those solutions with the fewest rings.

The pictures on the wall will have some characteristics upon which you may depend:

- Each picture is at least two feet wide.
- Each picture is at least two feet away from the left wall, the right wall, the floor, and the ceiling.
- For each picture, the floor-to-ceiling region that spans from two feet to the left of the picture to two feet to the right of the picture is unobstructed by other pictures.

Input

All distances are measured in feet. The first line of input contains three integers, N, S, and W, where $1 \leq N \leq 6$ is the number of pictures on the wall, $1 \leq S \leq 99$ is the starting ring's center height above the floor, and $1 \le W \le 99$ is the width between the left and right walls. The center of the starting disk will be 0.5 feet from the left wall.

The second line contains picture position information for the N pictures in order from the left side of the room to the right. Each picture is given by four integer distances in feet, $L\ R\ B\ T$. Here L and R are the distances from the left end of the wall to the left and right edges of the picture, and B and T are the distances from the floor to the bottom and top edges of the picture. For each picture $L + 2 \le R$ and $2 \le B < T$. For the leftmost picture, $2 \le L$. For the rightmost picture, $R \le W - 2$. The R distance for one picture is always at least 2 less than the L distance for the next picture to the right.

Assume the ceiling of the room is at least two feet higher than the center of the initial ring and the tops of all the pictures, so it is out of the way, and its exact height is not relevant.

Output

Output is a single floating-point number: $M + d$, where M is the minimal number of rings in a string ending with rightmost point less than 1 foot from the right wall, and d is the minimal distance from the right wall to the rightmost point on the Mth ring. The number that you output should be accurate to within an absolute or relative error of 10^{-3}.

Note that the form of the output is such that there is no special roundoff issue if you have a ring's right point at a distance very close to 1 foot from the right wall: With a distance 1 or over, there is room for 1 more ring, and the remaining distance to the wall goes down by 1 to compensate.

Sample Input 1	Sample Output 1
2 10 12 4 6 8 16 8 10 2 9	13.573069918537234

Problem D
No Duplicates

There is a game in which you try not to repeat a word while your opponent tries to see if you have repeated one.

"THE RAIN IN SPAIN" has no repeats.

"IN THE RAIN AND THE SNOW" repeats THE.

"THE RAIN IN SPAIN IN THE PLAIN" repeats THE and IN.

Write a program to test a phrase.

Input

Input is a line containing words separated by single spaces, where a word consists of one or more uppercase letters. A line contains no more than 80 characters.

Output

The output is "yes" if no word is repeated, and "no" if one or more words repeat.

Sample Input 1	Sample Output 1
`THE RAIN IN SPAIN`	`yes`

Sample Input 2	Sample Output 2
`IN THE RAIN AND THE SNOW`	`no`

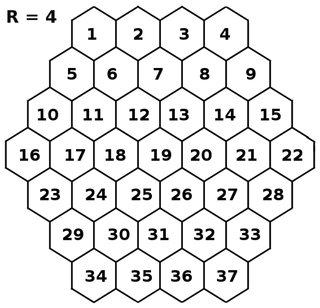
Sample Input 3	Sample Output 3
`THE RAIN IN SPAIN IN THE PLAIN`	`no`

This page is intentionally left (almost) blank.

Problem E
Honey Heist

0x67 is a scout ant searching for food and discovers a beehive nearby. As it approaches the honeycomb, 0x67 can sense an area inside packed with dried honey that can be easily carried back to the nest and stored for winter. However, it must burrow through the honeycomb to reach the cell containing the sweet loot. If 0x67 can create a passage to the honey to help the other ants find it, it will do so before returning to the nest.

The cells of the honeycomb are numbered in row major order, so cell IDs can be assigned as shown below:



When 0x67 discovers the opening to the honeycomb, it enters the cell. Some ants are stronger than others, depending on their age, so 0x67 can only chew through at most N cells before its jaw wears out and must return to the nest to recuperate. The honeycomb is hexagonal, and each edge length is R cells. 0x67 enters through a hole at location A and must get to the honey at location B by chewing a path through no more than N adjacent cells. Because ants can be competitive, 0x67 wants to reach the honey by chewing through the fewest possible cells. 0x67 can also sense some of the cells are hardened with wax and impossible to penetrate, so it will have to chew around those to reach the cell at location B.

Scout ants have rudimentary computational skills, and before 0x67 begins to chew, it will work out where it needs to go, and compute K, the least number of cells it needs to chew through to get from A to B, where B is the Kth cell. If $K > N$, 0x67 will not be strong enough to make the tunnel. When 0x67 returns to the nest, it will communicate to its nestmates how many cells it chewed through to get to B, or will report that it could not get to the honey.

Input

The input contains two lines. The first line contains five blank separated integers: $R\ N\ A\ B\ X$

R: the length (number of cells) of each edge of the grid, where $2 \le R \le 20$. The total number of cells in the grid can be determined by taking a difference of cubes, $R^3 - (R-1)^3$.

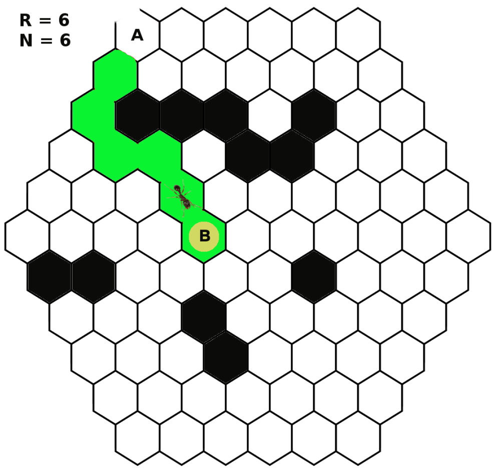


Figure E.1: K=6

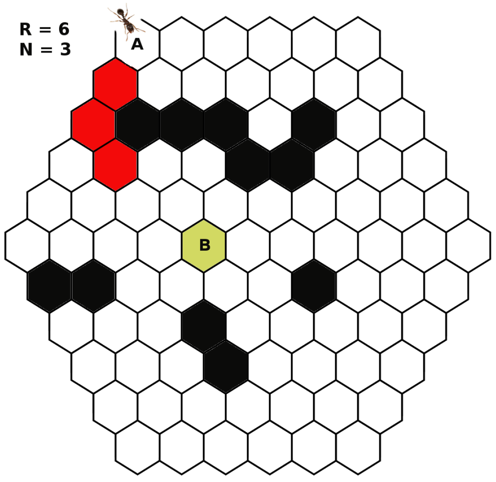


Figure E.2: No

N: the maximum number of cells 0x67 can chew through, where $1 \le N < R^3 - (R-1)^3$.

A: the starting cell ID, This cell is located on one of the grid edges: The cell has fewer than six neighbors.

B: the cell ID of the cell containing the honey, where $1 \le B \le R^3 - (R-1)^3$.

X: the number of wax-hardened cells, where $0 \le X < (R^3 - (R-1)^3) - 1$.

The second line contains X integers separated by spaces, where each integer is the ID of a wax-hardened cell.

The ID's, A, B, and all the ID's on the second line, are distinct positive integers less than or equal to $R^3 - (R-1)^3$.

Output

A single integer K if 0x67 reached the honey at cell B, where B is the Kth cell, otherwise the string `No` if it was impossible to reach the honey by chewing through N cells or less.

Sample Input 1

```
6 6 1 45 11
15 16 17 19 26 27 52 53 58 65 74
```

Sample Output 1

```
6
```

Sample Input 2

```
6 3 1 45 11
15 16 17 19 26 27 52 53 58 65 74
```

Sample Output 2

```
No
```

This page is intentionally left (almost) blank.

Problem F
Orderly Class

Ms. Thomas is managing her class of n students.

She placed all her students in a line, and gave the i-th student from the left a card with the letter a_i written on it.

She would now like to rearrange the students so that the i-th student from the left has a card with the letter b_i written on it.

To do this, she will choose some consecutive group of students, and reverse their order. Students will hold on to their original cards during this process.

She's now wondering, what is the number of valid ways to do this? (It may be impossible, in which case, the answer is zero).

With sequences $abba$ and $aabb$, Ms. Thomas can choose the group $a(bba)$. With sequences $caxcab$ and $cacxab$, Ms. Thomas can choose $ca(xc)ab$ or $c(axca)b$. With sequences a and z, there are clearly no solutions.

Input

The input is two lines of lowercase letters, A and B. The i-th character of A and B represent a_i and b_i respectively. It is guaranteed that A and B have the same positive length, and A and B are not identical. The common length is allowed to be as large as $100\,000$.

Output

For each test case, output a single integer, the number of ways Ms. Thomas can reverse some consecutive group of A to form the line specified by string B.

Sample Input 1

```
abba
aabb
```

Sample Output 1

```
1
```

Sample Input 2

```
caxcab
cacxab
```

Sample Output 2

```
2
```

Sample Input 3

```
a
z
```

Sample Output 3

```
0
```

Problem G
Faulty Robot

As part of a CS course, Alice just finished programming her robot to explore a graph having n nodes, labeled $1, 2, \ldots, n$, and m directed edges. Initially the robot starts at node 1.

While nodes may have several outgoing edges, Alice programmed the robot so that any node may have a *forced move* to a specific one of its neighbors. For example, it may be that node 5 has outgoing edges to neighbors 1, 4, and 6 but that Alice programs the robot so that if it leaves 5 it must go to neighbor 4.

If operating correctly, the robot will always follow forced moves away from a node, and if reaching a node that does not have a forced move, the robot stops. Unfortunately, the robot is a bit buggy, and it might violate those rules and move to a randomly chosen neighbor of a node (whether or not there had been a designated forced move from that node). However, such a bug will occur *at most once* (and might never happen).

Alice is having trouble debugging the robot, and would like your help to determine what are the possible nodes where the robot could stop and not move again.

We consider two sample graphs, as given in Figures G.1 and G.2. In these figures, a red arrow indicate an edge corresponding to a forced move, while black arrows indicate edges to other neighbors. The circle around a node is red if it is a possible stopping node.

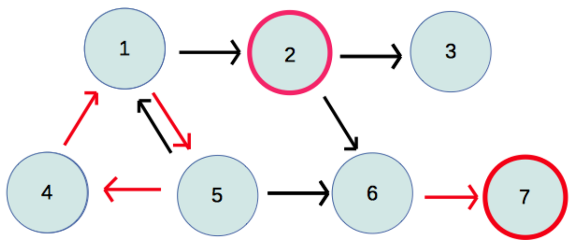


Figure G.1: First sample graph.

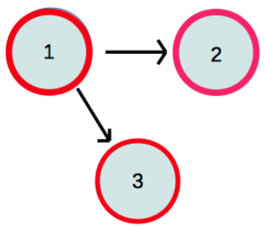


Figure G.2: Second sample graph.

In the first example, the robot will cycle forever through nodes 1, 5, and 4 if it does not make a buggy move. A bug could cause it to jump from 1 to 2, but that would be the only buggy move, and so it would never move on from there. It might also jump from 5 to 6 and then have a forced move to end at 7.

In the second example, there are no forced moves, so the robot would stay at 1 without any buggy moves. It might also make a buggy move from 1 to either 2 or 3, after which it would stop.

Input

The first line contains two integers n and m, designating the number of nodes and number of edges such that $1 \leq n \leq 10^3$, $0 \leq m \leq 10^4$. The next m lines will each have two integers a and b, $1 \leq |a|, b \leq n$ and $|a| \neq b$. If $a > 0$, there is a directed edge between nodes a and b that is not forced. If $a < 0$, then there is a forced directed edge from $-a$ to b. There will be at most 900 such forced moves. No two directed edges will be the same. No two starting nodes for forced moves will be the same.

Output

Display the *number* of nodes at which the robot might come to a rest.

Sample Input 1

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

Sample Output 1

```
2
```

Sample Input 2

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

Sample Output 2

```
3
```

Problem H
Hopscotch

You're playing hopscotch! You start at the origin and your goal is to hop to the lattice point (N, N). A hop consists of going from lattice point (x_1, y_1) to (x_2, y_2), where $x_1 < x_2$ and $y_1 < y_2$.

You dislike making small hops though. You've decided that for every hop you make between two lattice points, the x-coordinate must increase by at least X and the y-coordinate must increase by at least Y.

Compute the number of distinct paths you can take between $(0, 0)$ and (N, N) that respect the above constraints. Two paths are distinct if there is some lattice point that you visit in one path which you don't visit in the other.

Hint: The output involves arithmetic mod $10^9 + 7$. Note that with p a prime like $10^9 + 7$, and x an integer not equal to 0 mod p, then $x(x^{p-2})$ mod p equals 1 mod p.

Input

The input consists of a line of three integers, N X Y. You may assume $1 \leq X, Y \leq N \leq 10^6$.

Output

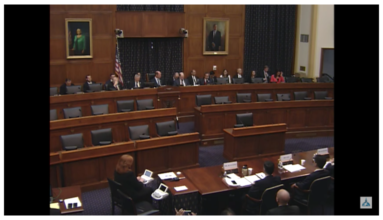
The number of distinct paths you can take between the two lattice points can be very large. Hence output this number modulo 1 000 000 007 ($10^9 + 7$).

Sample Input 1	Sample Output 1
`2 1 1`	`2`

Sample Input 2	Sample Output 2
`7 2 3`	`9`

This page is intentionally left (almost) blank.

Problem I
The Uncertainty of Politics



You have an upcoming trip to Washington D.C. and you are fascinated with the intricacies of Congressional committee hearings. You wish to attend as many hearings as possible during your trip, and your local representative has provided you with a pass that will get you into the audience of any hearing. But there are some challenges in planning your schedule. Specifically:

1. There are many committees, and thus many hearings, some of which take place at overlapping times.
2. While the committees are extremely punctual in terms of when to start a hearing, they are notoriously unpredictable in terms of how long the hearing lasts. Fortunately, rules do not actually allow for a filibuster of a committee hearing, so they cannot last forever.
3. It is considered rude to enter a hearing that is already underway, or to leave a hearing before it is complete. Given that you do not wish to embarrass the representative who provided your tickets, if you attend you must attend the entire hearing. Fortunately, hearings are very near to each other; as soon as one hearing is done, you can immediately join another hearing that is about to start.

Well in advance of your trip, Congress publishes a schedule of hearings, indicating for each one the time s at which the hearing will start, and then values a and b which represent, respectively, the shortest and longest possible length of that particular hearing. You are to assume that the actual length of the hearing will be a uniformly random *integer* over the inclusive interval $[a, b]$.

Your goal is to develop a strategy that maximizes the expected number of hearings that you can attend during your trip. As an example, consider a situation in which there are four hearings with parameters as follows:

hearing	s	a	b
Social media and elections	1	1	7
NASA missions	3	2	3
Oil and gas exploration	5	1	4
Hurricane recovery efforts	6	10	10

For this schedule, the optimal strategy will allow you to achieve an expected value of 2.125 hearings. To achieve this, you begin by attending the NASA hearing, which starts at time 3 and ends with equal probability at either time 5 or time 6 (given the hearing length that is uniformly distributed over $\{2, 3\}$). If the NASA hearing does end at time 5 you will immediately head to the oil and gas exploration hearing, and there is a $\frac{1}{4}$ chance that hearing will end at time 6, allowing you to make yet a third hearing (about hurricane recovery efforts). If the NASA hearing instead ends at time 6, you will go straight to the hurricane hearing.

By this strategy you will attend 3 hearings 12.5% of the time and 2 hearings the other 87.5% of the time, and thus expected value of 2.125. Note that if you were to start by attending the social media and elections hearing, you might optimistically make four hearings. However, a careful analysis will demonstrate that if you attend the first hearing, your optimal expected value is only 2.10714.

Input

The input begins with an integer n that designates the total number of scheduled hearings ($1 \leq n \leq 10^4$). Following that are n lines, each containing three integers s, a, and b, respectively representing the start time, minimum length, and maximum length of a hearing, such that $1 \leq s \leq 10^6$ and $1 \leq a \leq b \leq 10^6$. The hearings will be listed in nondecreasing order of their start times.

Output

Display the expected number of hearings of an optimal strategy. Your answer should have an absolute or relative error of at most 10^{-3}.

Sample Input 1	Sample Output 1

4
1 1 7
3 2 3
5 1 4
6 10 10
	2.125

Sample Input 2	Sample Output 2

5
1 1 7
1 1 6
3 2 3
5 1 4
6 10 10
	2.29166667