Task Overview Sheet

	Linear Garden	Teleporters	Pyramid Base
Type	Batch (stdin/stdout)*	Batch (stdin/stdout)*	Batch (stdin/stdout)*
Time Limit (per test case)	1.5 seconds	1 second	5 seconds
Memory Limit (per test case)	64 MB	64 MB	256 MB
Points	100	100	100

* C++ programmers should be aware that usage of C++ streams (cin/cout) may lead to I/O bottlenecks. We highly recommend usage of scanf/printf instead.

LINEAR GARDEN

Ramesses II has just returned victorious from battle. To commemorate his victory, he has decided to build a majestic garden. The garden will contain a long line of plants that will run all the way from his palace at Luxor to the temple of Karnak. It will consist only of lotus plants and papyrus plants, since they symbolize Upper and Lower Egypt respectively.

The garden must contain exactly ***N*** plants. Also, it must be balanced: in any contiguous section of the garden, the numbers of lotus and papyrus plants must not differ by more than 2.

A garden can be represented as a string of letters 'L' (lotus) and 'P' (papyrus). For example, for ***N=5*** there are 14 possible balanced gardens. In alphabetical order, these are: LLPLP, LLPPL, LPLLP, LPLPL, LPLPP, LPPLL, LPPLP, PLLPL, PLLPP, PLPLL, PLPLP, PLPPL, PPLLP, and PPLPL.

The possible balanced gardens of a certain length can be ordered alphabetically, and then numbered starting from 1. For example, for ***N=5***, garden number 12 is the garden PLPPL.

TASK

Write a program that, given the number of plants ***N*** and a string that represents a balanced garden, calculates the number assigned to this garden **modulo some given integer *M*.**

Note that for solving the task, the value of ***M*** has no importance other than simplifying computations.

CONSTRAINTS

1 <= ***N*** <= 1,000,000
7 <= ***M*** <= 10,000,000

GRADING

In inputs worth a total of 40 points, ***N*** will not exceed 40.

INPUT

Your program must read from the standard input the following data:

- Line 1 contains the integer ***N***, the number of plants in the garden.
- Line 2 contains the integer ***M***.
- Line 3 contains a string of ***N*** characters 'L' (lotus) or 'P' (papyrus) that represents a balanced garden.

OUTPUT

Your program must write to the standard output a single line containing one integer between 0 and ***M-1*** (inclusive), the number assigned to the garden described in the input, **modulo *M***.

EXAMPLE

Sample input 1	Sample output 1
5 7 PLPPL	5

The actual number assigned to PLPPL is **12**. So, the output is **12** modulo **7,** which is **5**.

Sample input 2	Sample output 2
12 10000 LPLLPLPPLPLL	39

TELEPORTERS

You are participating in a competition that involves crossing Egypt from west to east along a straight line segment. Initially you are located at the westmost point of the segment. It is a rule of the competition that you must always move along the segment, and always eastward.

There are ***N*** teleporters on the segment. A teleporter has two endpoints. Whenever you reach one of the endpoints, the teleporter immediately teleports you to the other endpoint. (Note that, depending on which endpoint of the teleporter you reach, teleportation can transport you either eastward or westward of your current position.) After being teleported, you must continue to move eastward along the segment; you can never avoid a teleporter endpoint that is on your way. There will never be two teleporter endpoints at the same position. Endpoints will be strictly between the start and the end of the segment.

Every time you get teleported, you earn 1 point. The objective of the competition is to earn as many points as possible. In order to maximize the points you earn, you are allowed to add up to ***M*** new teleporters to the segment before you start your journey. You also earn points for using the new teleporters.

You can set the endpoints of the new teleporters wherever you want (even at non-integer coordinates) as long as they do not occupy a position already occupied by another endpoint. That is, the positions of the endpoints of all teleporters must be unique. Also, endpoints of new teleporters must lie strictly between the start and the end of the segment.

Note that it is guaranteed that no matter how you add the teleporters, you can always reach the end of the segment.

TASK

Write a program that, given the position of the endpoints of the ***N*** teleporters, and the number ***M*** of new teleporters that you can add, computes the maximum number of points you can earn.

CONSTRAINTS

$1 <= N <= 1{,}000{,}000$ The number of teleporters initially on the segment.

$1 <= M <= 1{,}000{,}000$ The maximum number of new teleporters you can add.

$1 <= W_X < E_X <= 2{,}000{,}000$ The distances from the beginning of the segment to the western and eastern endpoints of teleporter X.

INPUT

Your program must read from the standard input the following data:

- Line 1 contains the integer ***N***, the number of teleporters initially on the segment.
- Line 2 contains the integer ***M***, the maximum number of new teleporters that you can add.
- Each of the next ***N*** lines describes one teleporter. The i^{th} of these lines describes the i^{th} teleporter. Each line consists of 2 integers: W_i and E_i separated by a space. They represent respectively the distances from the beginning of the segment to the western and eastern endpoints of the teleporter.

No two endpoints of the given teleporters share the same position. The segment that you will be travelling on starts at position 0 and ends at position 2,000,001.

OUTPUT

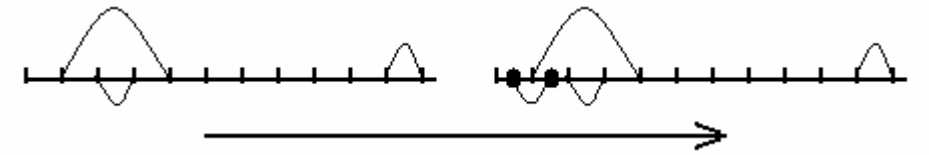
Your program must write to the standard output a single line containing one integer, the maximum number of points you can earn.

GRADING

In test data worth 30 points, ***N*** <= 500 and ***M*** <= 500.

EXAMPLE

Sample input 1	Sample output 1
3 1 10 11 1 4 2 3	6



The first figure shows a segment with the three original teleporters. The second figure shows the same segment after adding a new teleporter with endpoints at 0.5 and at 1.5.

After adding the new teleporter as shown in the figure, your travel would be the following:

- You start at position 0, moving eastward.
- You reach the endpoint at position 0.5 and get teleported to position 1.5 (you earn 1 point).
- You continue to move east and reach endpoint at position 2; you get teleported to position 3 (you have 2 points).
- You reach endpoint at position 4, and get teleported to 1 (you have 3 points).
- You reach endpoint at 1.5, and get teleported to 0.5 (you have 4 points).
- You reach endpoint at 1, and get teleported to 4 (you have 5 points).
- You reach endpoint at 10, and get teleported to 11 (you have 6 points).

- You continue until you reach the end of the segment finishing with a total score of 6 points.

Sample input 2	Sample output 2
3 3 5 7 6 10 1999999 2000000	12

Pyramid Base

You have been asked to find the largest affordable location for constructing a new pyramid. In order to help you decide, you have been provided with a survey of the available land which has been conveniently divided into an ***M*** by ***N*** grid of square cells. The base of the pyramid must be a square with sides parallel to those of the grid.

The survey has identified a set of ***P*** possibly overlapping obstacles, which are described as rectangles in the grid with sides parallel to those of the grid. In order to build the pyramid, all the cells covered by its base must be cleared of any obstacles. Removing the i^{th} obstacle has a cost C_i. Whenever an obstacle is removed, it must be removed completely, that is, you cannot remove only part of an obstacle. Also, please note that removing an obstacle does not affect any other obstacles that overlap it.

TASK

Write a program that, given the dimensions ***M*** and ***N*** of the survey, the description of the ***P*** obstacles, the cost of removing each of the obstacles, and the budget ***B*** you have, finds the maximum possible side length of the base of the pyramid such that the total cost of removing obstacles does not exceed ***B***.

CONSTRAINTS AND GRADING

Your program will be graded on three disjoint sets of tests. For all of them, the following constraints apply:

$1 \le M, N \le 1{,}000{,}000$	The dimensions of the grid.
$1 \le C_i \le 7{,}000$	The cost of removing the i^{th} obstacle.
$1 \le X_{i1} \le X_{i2} \le M$	X coordinates of the leftmost and the rightmost cells of the i^{th} obstacle.
$1 \le Y_{i1} \le Y_{i2} \le N$	Y coordinates of the bottommost and the topmost cells of the i^{th} obstacle.

<u>In the first set of tests worth 35 points:</u>

$B = 0$	The budget you have. (You cannot remove any obstacles.)
$1 \le P \le 1{,}000$	The number of obstacles in the grid.

<u>In the second set of tests worth 35 points:</u>

$0 < B \le 2{,}000{,}000{,}000$	The budget you have.
$1 \le P \le 30{,}000$	The number of obstacles in the grid.

<u>In the third set of tests worth 30 points:</u>

$B = 0$	The budget you have. (You cannot remove any obstacles.)
$1 \le P \le 400{,}000$	The number of obstacles in the grid.

INPUT

Your program must read from the standard input the following data:

- Line 1 contains two integers separated by a single space that represent ***M*** and ***N*** respectively.
- Line 2 contains the integer ***B***, the maximum cost you can afford (i.e., your budget).
- Line 3 contains the integer ***P***, the number of obstacles found in the survey.
- Each of the next ***P*** lines describes an obstacle. The i^{th} of these lines describes the i^{th} obstacle. Each line consists of 5 integers: X_{i1}, Y_{i1}, X_{i2}, Y_{i2}, and C_i separated by single spaces. They represent respectively the coordinates of the bottommost leftmost cell of the obstacle, the coordinates of the topmost rightmost cell of the obstacle, and the cost of removing the obstacle. The bottommost leftmost cell on the grid has coordinates (1, 1) and the topmost rightmost cell has coordinates (***M***, ***N***).

OUTPUT

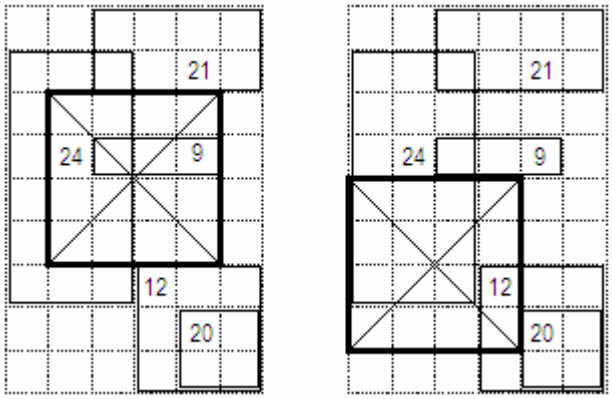
Your program must write to the standard output a single line containing one integer, the maximum possible side length of the base of the pyramid that can be prepared. If it is not possible to build any pyramid, your program should output the number 0.

DETAILED FEEDBACK

During the contest, your submissions for this task will be evaluated on some of the official test data showing you a summary of the results.

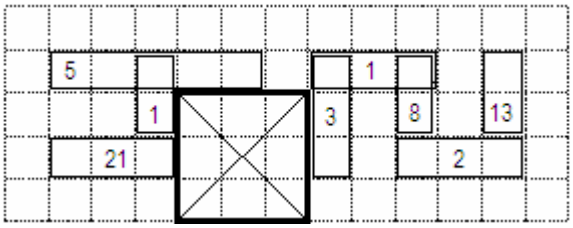
EXAMPLE

Sample input 1	Sample output 1
6 9 42 5 4 1 6 3 12 3 6 5 6 9 1 3 3 8 24 3 8 6 9 21 5 1 6 2 20	4



The figure shows two possible locations for the pyramid's base, both having a side of length 4.

Sample input 2	Sample output 2
13 5 0 8 8 4 10 4 1 4 3 4 4 1 10 2 12 2 2 8 2 8 4 3 2 4 6 4 5 10 3 10 4 8 12 3 12 4 13 2 2 4 2 21	3



The figure shows the only possible location for the pyramid's base having a side of length 3.