Problem A
Secret of Chocolate Poles

Time Limit: 1 second

Wendy, the master of a chocolate shop, is thinking of displaying poles of chocolate disks in the showcase. She can use three kinds of chocolate disks: white thin disks, dark thin disks, and dark thick disks. The thin disks are 1 cm thick, and the thick disks are k cm thick. Disks will be piled in glass cylinders.

Each pole should satisfy the following conditions for her secret mission, which we cannot tell.

- A pole should consist of at least one disk.
- The total thickness of disks in a pole should be less than or equal to l cm.
- The top disk and the bottom disk of a pole should be dark.
- A disk directly upon a white disk should be dark and vice versa.

As examples, six side views of poles are drawn in Figure A.1. These are the only possible side views she can make when $l = 5$ and $k = 3$.

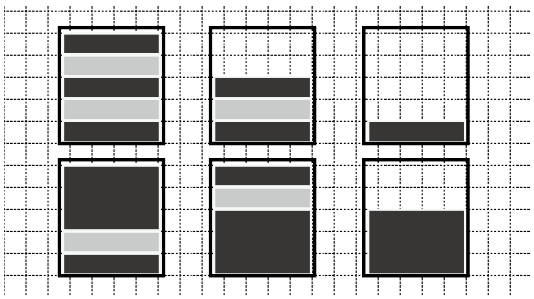


Figure A.1. Six chocolate poles corresponding to Sample Input 1

Your task is to count the number of distinct side views she can make for given l and k to help her accomplish her secret mission.

Input

The input consists of a single test case in the following format.

l k

Here, the maximum possible total thickness of disks in a pole is l cm, and the thickness of the thick disks is k cm. l and k are integers satisfying $1 \leq l \leq 100$ and $2 \leq k \leq 10$.

Output

Output the number of possible distinct patterns.

Sample Input 1	**Sample Output 1**
5 3	6

Sample Input 2	**Sample Output 2**
9 10	5

Sample Input 3	**Sample Output 3**
10 10	6

Sample Input 4	**Sample Output 4**
20 5	86

Sample Input 5	**Sample Output 5**
100 2	3626169232670

Problem B
Parallel Lines

Time Limit: 10 seconds

Given an even number of distinct planar points, consider coupling all of the points into pairs. All the possible couplings are to be considered as long as all the given points are coupled to one and only one other point.

When lines are drawn connecting the two points of all the coupled point pairs, some of the drawn lines can be parallel to some others. Your task is to find the maximum number of parallel line pairs considering all the possible couplings of the points.

For the case given in the first sample input with four points, there are three patterns of point couplings as shown in Figure B.1. The numbers of parallel line pairs are 0, 0, and 1, from the left. So the maximum is 1.

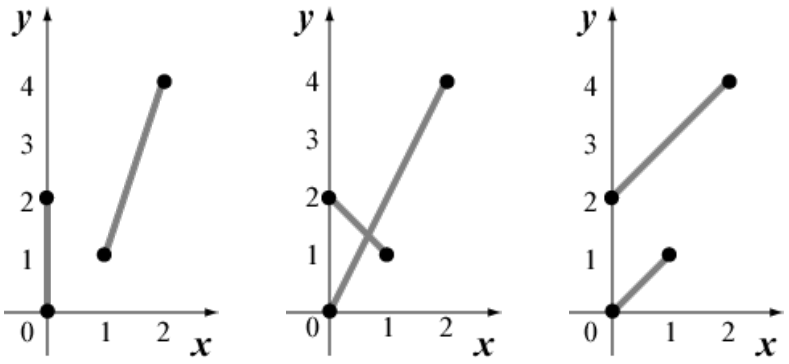


Figure B.1. All three possible couplings for Sample Input 1

For the case given in the second sample input with eight points, the points can be coupled as shown in Figure B.2. With such a point pairing, all four lines are parallel to one another. In other words, the six line pairs (L_1, L_2), (L_1, L_3), (L_1, L_4), (L_2, L_3), (L_2, L_4) and (L_3, L_4) are parallel. So the maximum number of parallel line pairs, in this case, is 6.

Input

The input consists of a single test case of the following format.

$$
\begin{array}{l}
m \\
x_1\ y_1 \\
\vdots \\
x_m\ y_m
\end{array}
$$

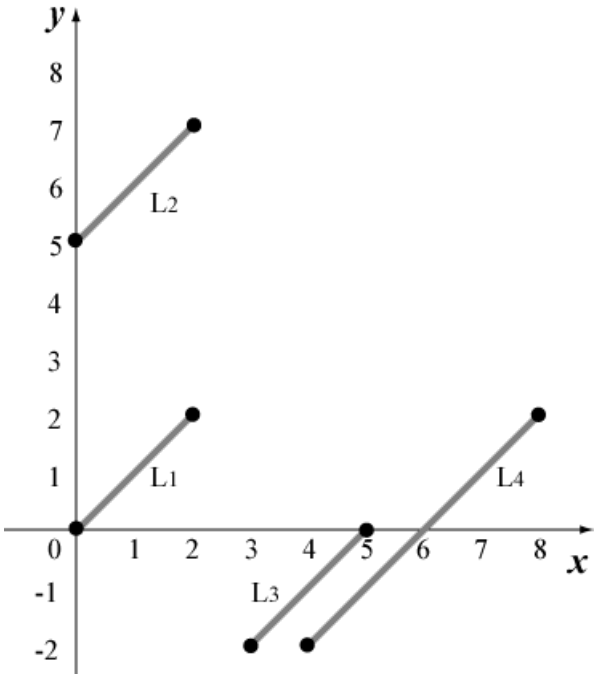


Figure B.2. Maximizing the number of parallel line pairs for Sample Input 2

The first line contains an even integer m, which is the number of points ($2 \leq m \leq 16$). Each of the following m lines gives the coordinates of a point. Integers x_i and y_i ($-1000 \leq x_i \leq 1000$, $-1000 \leq y_i \leq 1000$) in the i-th line of them give the x- and y-coordinates, respectively, of the i-th point.

The positions of points are all different, that is, $x_i \neq x_j$ or $y_i \neq y_j$ holds for all $i \neq j$. Furthermore, No three points lie on a single line.

Output

Output the maximum possible number of parallel line pairs stated above, in one line.

Sample Input 1	**Sample Output 1**
4 0 0 1 1 0 2 2 4	1

Sample Input 2

```
8
0 0
0 5
2 2
2 7
3 -2
5 0
4 -2
8 2
```

Sample Output 2

```
6
```

Sample Input 3

```
6
0 0
0 5
3 -2
3 5
5 0
5 7
```

Sample Output 3

```
3
```

Sample Input 4

```
2
-1000 1000
1000 -1000
```

Sample Output 4

```
0
```

Sample Input 5

```
16
327 449
-509 761
-553 515
360 948
147 877
-694 468
241 320
463 -753
-206 -991
473 -738
-156 -916
-215 54
-112 -476
-452 780
-18 -335
-146 77
```

Sample Output 5

```
12
```

Problem C
Medical Checkup

Time Limit: 2 seconds

Students of the university have to go for a medical checkup, consisting of lots of checkup items, numbered 1, 2, 3, and so on.

Students are now forming a long queue, waiting for the checkup to start. Students are also numbered 1, 2, 3, and so on, from the top of the queue. They have to undergo checkup items in the order of the item numbers, not skipping any of them nor changing the order. The order of students should not be changed either.

Multiple checkup items can be carried out in parallel, but each item can be carried out for only one student at a time. Students have to wait in queues of their next checkup items until all the others before them finish.

Each of the students is associated with an integer value called *health condition.* For a student with the health condition h, it takes h minutes to finish each of the checkup items. You may assume that no interval is needed between two students on the same checkup item or two checkup items for a single student.

Your task is to find the items students are being checked up or waiting for at a specified time t.

Input

The input consists of a single test case in the following format.

$$
\begin{array}{l}
n\ t \\
h_1 \\
\vdots \\
h_n
\end{array}
$$

n and t are integers. n is the number of the students ($1 \leq n \leq 10^5$). t specifies the time of our concern ($0 \leq t \leq 10^9$). For each i, the integer h_i is the health condition of student i ($1 \leq h_i \leq 10^9$).

Output

Output n lines each containing a single integer. The i-th line should contain the checkup item number of the item which the student i is being checked up or is waiting for, at $(t+0.5)$ minutes after the checkup starts. You may assume that all the students are yet to finish some of the checkup items at that moment.

Sample Input 1	Sample Output 1
3 20 5 7 3	5 3 2

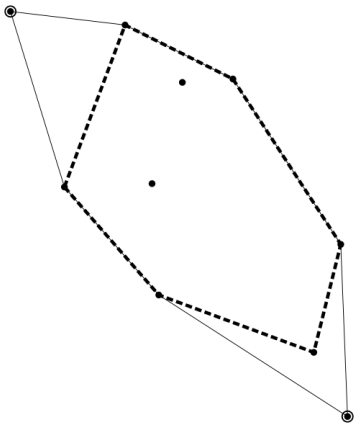
Sample Input 2	Sample Output 2
5 1000000000 5553 2186 3472 2605 1790	180083 180083 180082 180082 180082

Problem D
Making Perimeter of the Convex Hull Shortest

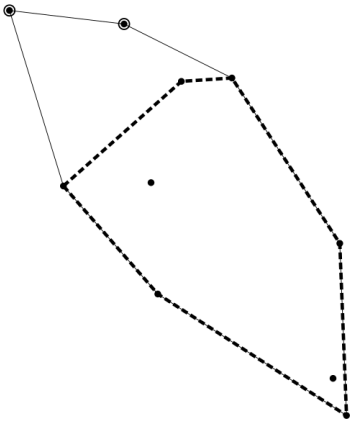
Time Limit: 10 seconds

The convex hull of a set of three or more planar points is, when not all of them are on one line, the convex polygon with the smallest area that has all the points of the set on its boundary or in its inside. Your task is, given positions of the points of a set, to find how much shorter the perimeter of the convex hull can be made by excluding *two* points from the set.

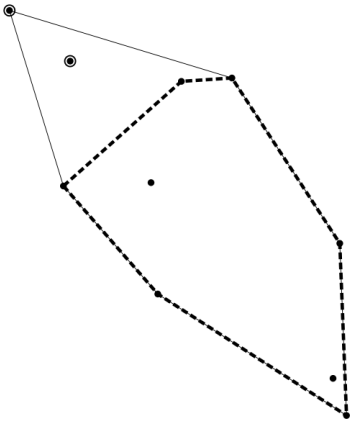
The figures below correspond to the three cases given as Sample Input 1 to 3. Encircled points are excluded to make the shortest convex hull depicted as thick dashed lines.



Sample Input 1



Sample Input 2



Sample Input 3

Input

The input consists of a single test case in the following format.

$$
\begin{array}{l}
n \\
x_1\ y_1 \\
\vdots \\
x_n\ y_n
\end{array}
$$

Here, n is the number of points in the set satisfying $5 \leq n \leq 10^5$. For each i, (x_i, y_i) gives the coordinates of the position of the i-th point in the set. x_i and y_i are integers between -10^6 and 10^6, inclusive. All the points in the set are distinct, that is, $x_j \neq x_k$ or $y_j \neq y_k$ holds when $j \neq k$. It is guaranteed that no single line goes through $n - 2$ or more points in the set.

Output

Output the difference of the perimeter of the convex hull of the original set and the shortest of the perimeters of the convex hulls of the subsets with two points excluded from the original set. The output should not have an error greater than 10^{-4}.

Sample Input 1	Sample Output 1
10 -53 62 -19 58 -11 11 -9 -22 45 -7 37 -39 47 -58 -2 41 -37 10 13 42	72.96316928

Sample Input 2	Sample Output 2
10 -53 62 -19 58 -11 11 -9 -22 45 -7 43 -47 47 -58 -2 41 -37 10 13 42	62.62947992

Sample Input 3	Sample Output 3
10 -53 62 -35 47 -11 11 -9 -22 45 -7 43 -47 47 -58 -2 41 -37 10 13 42	61.58166534

Problem E
Black or White

Time Limit: 2 seconds

Here lies a row of a number of bricks each painted either black or white. With a single stroke of your brush, you can overpaint a part of the row of bricks at once with either black or white paint. Using white paint, all the black bricks in the painted part become white while originally white bricks remain white; with black paint, white bricks become black and black ones remain black. The number of bricks painted in one stroke, however, is limited because your brush cannot hold too much paint at a time. For each brush stroke, you can paint any part of the row with any number of bricks up to the limit.

In the first case of the sample input, the initial colors of four bricks are black, white, white, and black. You can repaint them to white, black, black, and white with two strokes: the first stroke paints all four bricks white and the second stroke paints two bricks in the middle black.

Your task is to calculate the minimum number of brush strokes needed to change the brick colors as specified. Never mind the cost of the paints.

Input

The input consists of a single test case formatted as follows.

$n\ k$
s
t

The first line contains two integers n and k ($1 \leq k \leq n \leq 500\ 000$). n is the number of bricks in the row and k is the maximum number of bricks painted in a single stroke. The second line contains a string s of n characters, which indicates the initial colors of the bricks. The third line contains another string t of n characters, which indicates the desired colors of the bricks. All the characters in both s and t are either `B` or `W` meaning black and white, respectively.

Output

Output the minimum number of brush strokes required to repaint the bricks into the desired colors.

Sample Input 1	**Sample Output 1**
4 4 BWWB WBBW	2

Sample Input 2

```
4 3
BWWB
WBBW
```

Sample Output 2

```
3
```

Sample Input 3

```
4 3
BWWW
BWWW
```

Sample Output 3

```
0
```

Sample Input 4

```
7 1
BBWBWBW
WBBWWBB
```

Sample Output 4

```
4
```

Problem F
Pizza Delivery

Time Limit: 2 seconds

Alyssa is a college student, living in New Tsukuba City. All the streets in the city are one-way. A new social experiment starting tomorrow is on alternative traffic regulation reversing the one-way directions of street sections. Reversals will be on one single street section between two adjacent intersections for each day; the directions of all the other sections will not change, and the reversal will be canceled on the next day.

Alyssa orders a piece of pizza everyday from the same pizzeria. The pizza is delivered along the shortest route from the intersection with the pizzeria to the intersection with Alyssa's house.

Altering the traffic regulation may change the shortest route. Please tell Alyssa how the social experiment will affect the pizza delivery route.

Input

The input consists of a single test case in the following format.

$$
\begin{array}{l}
n\ m \\
a_1\ b_1\ c_1 \\
\vdots \\
a_m\ b_m\ c_m
\end{array}
$$

The first line contains two integers, n, the number of intersections, and m, the number of street sections in New Tsukuba City ($2 \leq n \leq 100\,000$, $1 \leq m \leq 100\,000$). The intersections are numbered 1 through n and the street sections are numbered 1 through m.

The following m lines contain the information about the street sections, each with three integers a_i, b_i, and c_i ($1 \leq a_i \leq n$, $1 \leq b_i \leq n$, $a_i \neq b_i$, $1 \leq c_i \leq 100\,000$). They mean that the street section numbered i connects two intersections with the one-way direction from a_i to b_i, which will be reversed on the i-th day. The street section has the length of c_i. Note that there may be more than one street section connecting the same pair of intersections.

The pizzeria is on the intersection 1 and Alyssa's house is on the intersection 2. It is guaranteed that at least one route exists from the pizzeria to Alyssa's before the social experiment starts.

Output

The output should contain m lines. The i-th line should be

- `HAPPY` if the shortest route on the i-th day will become shorter,
- `SOSO` if the length of the shortest route on the i-th day will not change, and
- `SAD` if the shortest route on the i-th day will be longer or if there will be no route from the pizzeria to Alyssa's house.

Alyssa doesn't mind whether the delivery bike can go back to the pizzeria or not.

Sample Input 1

```
4 5
1 3 5
3 4 6
4 2 7
2 1 18
2 3 12
```

Sample Output 1

```
SAD
SAD
SAD
SOSO
HAPPY
```

Sample Input 2

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

Sample Output 2

```
SOSO
SAD
SOSO
SAD
SOSO
```

Sample Input 3

```
10 14
1 7 9
1 8 3
2 8 4
2 6 11
3 7 8
3 4 4
3 2 1
3 2 7
4 8 4
5 6 11
5 8 12
6 10 6
7 10 8
8 3 6
```

Sample Output 3

```
SOSO
SAD
HAPPY
SOSO
SOSO
SOSO
SAD
SOSO
SOSO
SOSO
SOSO
SOSO
SOSO
SAD
```

Problem G
Rendezvous on a Tetrahedron

Time Limit: 1 second

One day, you found two worms P and Q crawling on the surface of a regular tetrahedron with four vertices A, B, C, and D. Both worms started from the vertex A, went straight ahead, and stopped crawling after a while.

When a worm reached one of the edges of the tetrahedron, it moved on to the adjacent face and kept going without changing the angle to the crossed edge (Figure G.1).

Write a program which tells whether or not P and Q were on the same face of the tetrahedron when they stopped crawling.

You may assume that each of the worms is a point without length, area, or volume.

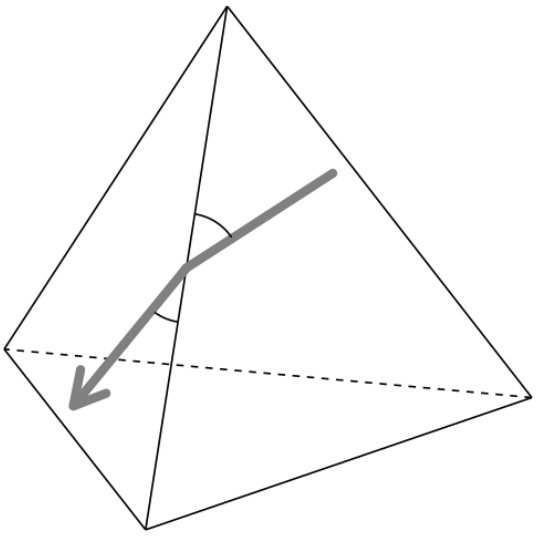


Figure G.1. Crossing an edge

Incidentally, lengths of the two trails the worms left on the tetrahedron were exact integral multiples of the unit length. Here, the unit length is the edge length of the tetrahedron. Each trail is more than 0.001 unit distant from any vertices, except for its start point and its neighborhood. This means that worms have crossed at least one edge. Both worms stopped at positions more than 0.001 unit distant from any of the edges.

The initial crawling direction of a worm is specified by two items: the edge XY which is the first edge the worm encountered after its start, and the angle d between the edge AX and the

direction of the worm, in degrees.

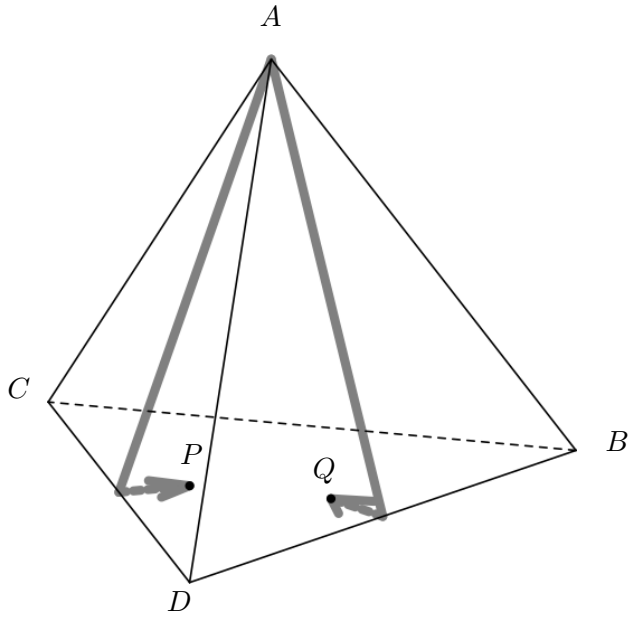


Figure G.2. Trails of the worms corresponding to Sample Input 1

Figure G.2 shows the case of Sample Input 1. In this case, P went over the edge CD and stopped on the face opposite to the vertex A, while Q went over the edge DB and also stopped on the same face.

Input

The input consists of a single test case, formatted as follows.

$X_P Y_P \ \ d_P \ \ l_P$
$X_Q Y_Q \ \ d_Q \ \ l_Q$

$X_W Y_W$ ($W = P, Q$) is the first edge the worm W crossed after its start. $X_W Y_W$ is one of `BC`, `CD` or `DB`.

An integer d_W ($1 \leq d_W \leq 59$) is the angle in degrees between edge AX_W and the initial direction of the worm W on the face $\triangle AX_W Y_W$.

An integer l_W ($1 \leq l_W \leq 20$) is the length of the trail of worm W left on the surface, in unit lengths.

Output

Output `YES` when and only when the two worms stopped on the same face of the tetrahedron. Otherwise, output `NO`.

Sample Input 1	Sample Output 1
`CD 30 1` `DB 30 1`	`YES`

Sample Input 2	Sample Output 2
`BC 1 1` `DB 59 1`	`YES`

Sample Input 3	Sample Output 3
`BC 29 20` `BC 32 20`	`NO`

Problem H
Homework

Time Limit: 2 seconds

Taro is a student of Ibaraki College of Prominent Computing. In this semester, he takes two courses, mathematics and informatics. After each class, the teacher may assign homework. Taro may be given multiple assignments in a single class, and each assignment may have a different deadline. Each assignment has a unique ID number.

Everyday after school, Taro completes at most one assignment as follows. First, he decides which course's homework to do at random by flipping a coin. Let S be the set of all the unfinished assignments of the chosen course whose deadline has not yet passed. If S is empty, he plays a video game without doing any homework on that day even if there are unfinished assignments of the other course. Otherwise, with $T \subseteq S$ being the set of assignments with the nearest deadline among S, he completes the one with the smallest assignment ID among T.

The number of assignments Taro will complete until the end of the semester depends on the result of coin flips. Given the schedule of homework assignments, your task is to compute the maximum and the minimum numbers of assignments Taro will complete.

Input

The input consists of a single test case in the following format.

$$
\begin{aligned}
&n\ m\\
&s_1\ t_1\\
&\vdots\\
&s_n\ t_n
\end{aligned}
$$

The first line contains two integers n and m satisfying $1 \leq m < n \leq 400$. n denotes the total number of assignments in this semester, and m denotes the number of assignments of the mathematics course (so the number of assignments of the informatics course is $n - m$). Each assignment has a unique ID from 1 to n; assignments with IDs 1 through m are those of the mathematics course, and the rest are of the informatics course. The next n lines show the schedule of assignments. The i-th line of them contains two integers s_i and t_i satisfying $1 \leq s_i \leq t_i \leq 400$, which means that the assignment of ID i is given to Taro on the s_i-th day of the semester, and its deadline is the end of the t_i-th day.

Output

In the first line, print the maximum number of assignments Taro will complete. In the second line, print the minimum number of assignments Taro will complete.

Sample Input 1

```
6 3
1 2
1 5
2 3
2 6
4 5
4 6
```

Sample Output 1

```
6
2
```

Sample Input 2

```
6 3
1 1
2 3
4 4
1 1
2 4
3 3
```

Sample Output 2

```
4
3
```

Problem I
Starting a Scenic Railroad Service

Time Limit: 2 seconds

Jim, working for a railroad company, is responsible for planning a new tourist train service. He is sure that the train route along a scenic valley will arise a big boom, but not quite sure how big the boom will be.

A market survey was ordered and Jim has just received an estimated list of passengers' travel sections. Based on the list, he'd like to estimate the minimum number of train seats that meets the demand.

Providing as many seats as all of the passengers may cost unreasonably high. Assigning the same seat to more than one passenger without overlapping travel sections may lead to a great cost cutback.

Two different policies are considered on seat assignments. As the views from the train windows depend on the seat positions, it would be better if passengers can choose a seat. One possible policy (named 'policy-1') is to allow the passengers to choose an arbitrary seat among all the remaining seats when they make their reservations. As the order of reservations is unknown, all the possible orders must be considered on counting the required number of seats.

The other policy (named 'policy-2') does not allow the passengers to choose their seats; the seat assignments are decided by the railroad operator, not by the passengers, after all the reservations are completed. This policy may reduce the number of the required seats considerably.

Your task is to let Jim know how different these two policies are by providing him a program that computes the numbers of seats required under the two seat reservation policies.

Let us consider a case where there are four stations, S1, S2, S3, and S4, and four expected passengers p_1, p_2, p_3, and p_4 with the travel list below.

passenger	from	to
p_1	S1	S2
p_2	S2	S3
p_3	S1	S3
p_4	S3	S4

The travel sections of p_1 and p_2 do not overlap, that of p_3 overlaps those of p_1 and p_2, and that of p_4 does not overlap those of any others.

Let's check if two seats would suffice under the policy-1. If p_1 books a seat first, either of the two seats can be chosen. If p_2 books second, as the travel section does not overlap that of p_1,

the same seat can be booked, but the other seat may look more attractive to p_2. If p_2 reserves a seat different from that of p_1, there will remain no available seats for p_3 between S1 and S3 (Figure I.1).

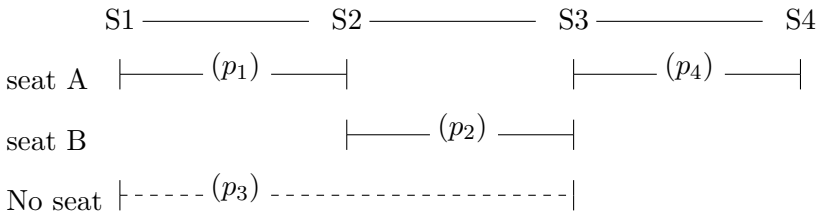


Figure I.1. With two seats

With three seats, p_3 can find a seat with any seat reservation combinations by p_1 and p_2. p_4 can also book a seat for there are no other passengers between S3 and S4 (Figure I.2).

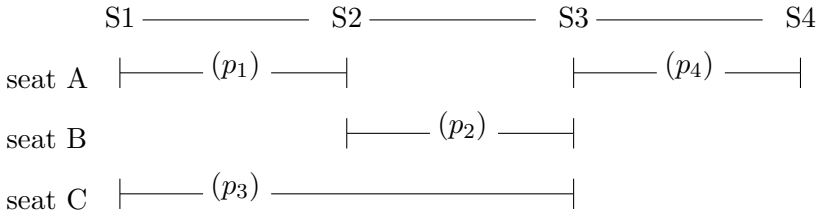


Figure I.2. With three seats

For this travel list, only three seats suffice considering all the possible reservation orders and seat preferences under the policy-1.

On the other hand, deciding the seat assignments after all the reservations are completed enables a tight assignment with only two seats under the policy-2 (Figure I.3).

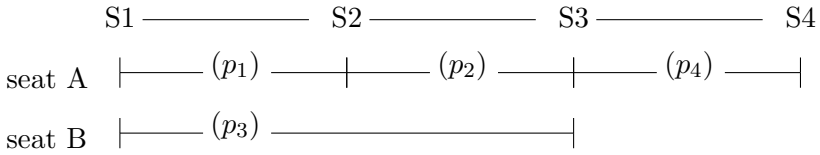


Figure I.3. Tight assignment to two seats

Input

The input consists of a single test case of the following format.

n
a_1 b_1
$\vdots$
a_n b_n

Here, the first line has an integer n, the number of the passengers in the estimated list of passengers' travel sections ($1 \leq n \leq 200\,000$). The stations are numbered starting from 1 in their order along the route. Each of the following n lines describes the travel for each passenger by two integers, the boarding and the alighting station numbers, a_i and b_i, respectively ($1 \leq a_i < b_i \leq 100\,000$). Note that more than one passenger in the list may have the same boarding and alighting stations.

Output

Two integers s_1 and s_2 should be output in a line in this order, separated by a space. s_1 and s_2 are the numbers of seats required under the policy-1 and -2, respectively.

Sample Input 1

```
4
1 3
1 3
3 6
3 6
```

Sample Output 1

```
2 2
```

Sample Input 2

```
4
1 2
2 3
1 3
3 4
```

Sample Output 2

```
3 2
```

Sample Input 3

```
10
84 302
275 327
364 538
26 364
29 386
545 955
715 965
404 415
903 942
150 402
```

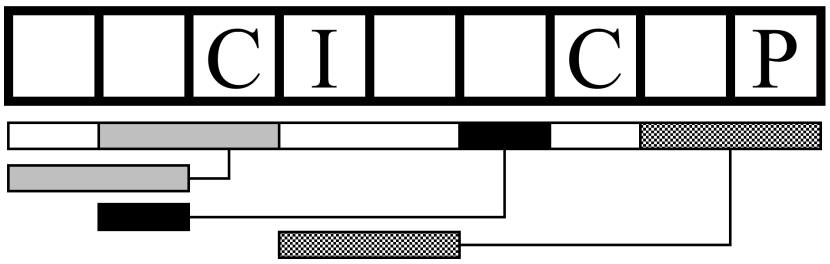
Sample Output 3

```
6 5
```

Problem J
String Puzzle
Time Limit: 2 seconds

Amazing Coding Magazine is popular among young programmers for its puzzle solving contests offering catchy digital gadgets as the prizes. The magazine for programmers naturally encourages the readers to solve the puzzles by writing programs. Let's give it a try!

The puzzle in the latest issue is on deciding some of the letters in a string (*the secret string,* in what follows) based on a variety of hints. The figure below depicts an example of the given hints.



The first hint is the number of letters in the secret string. In the example of the figure above, it is nine, and the nine boxes correspond to nine letters. The letter positions (boxes) are numbered starting from 1, from the left to the right.

The hints of the next kind simply tell the letters in the secret string at some specific positions. In the example, the hints tell that the letters in the 3rd, 4th, 7th, and 9th boxes are `C`, `I`, `C`, and `P`, respectively.

The hints of the final kind are on duplicated substrings in the secret string. The bar immediately below the boxes in the figure is partitioned into some sections corresponding to substrings of the secret string. Each of the sections may be connected by a line running to the left with another bar also showing an extent of a substring. Each of the connected pairs indicates that substrings of the two extents are identical. One of this kind of hints in the example tells that the letters in boxes 8 and 9 are identical to those in boxes 4 and 5, respectively. From this, you can easily deduce that the substring is `IP`.

Note that, not necessarily all of the identical substring pairs in the secret string are given in the hints; some identical substring pairs may not be mentioned.

Note also that two extents of a pair may overlap each other. In the example, the two-letter substring in boxes 2 and 3 is told to be identical to one in boxes 1 and 2, and these two extents share the box 2.

In this example, you can decide letters at all the positions of the secret string, which are "`CCCIPCCIP`". In general, the hints may not be enough to decide all the letters in the secret string.

The answer of the puzzle should be letters at the specified positions of the secret string. When the letter at the position specified cannot be decided with the given hints, the symbol `?` should be answered.

Input

The input consists of a single test case in the following format.

$$
\begin{array}{l}
n\ a\ b\ q \\
x_1\ c_1 \\
\vdots \\
x_a\ c_a \\
y_1\ h_1 \\
\vdots \\
y_b\ h_b \\
z_1 \\
\vdots \\
z_q
\end{array}
$$

The first line contains four integers n, a, b, and q. n ($1 \leq n \leq 10^9$) is the length of the secret string, a ($0 \leq a \leq 1000$) is the number of the hints on letters in specified positions, b ($0 \leq b \leq 1000$) is the number of the hints on duplicated substrings, and q ($1 \leq q \leq 1000$) is the number of positions asked.

The i-th line of the following a lines contains an integer x_i and an uppercase letter c_i meaning that the letter at the position x_i of the secret string is c_i. These hints are ordered in their positions, i.e., $1 \leq x_1 < \cdots < x_a \leq n$.

The i-th line of the following b lines contains two integers, y_i and h_i. It is guaranteed that they satisfy $2 \leq y_1 < \cdots < y_b \leq n$ and $0 \leq h_i < y_i$. When h_i is *not* 0, the substring of the secret string starting from the position y_i with the length $y_{i+1} - y_i$ (or $n+1-y_i$ when $i = b$) is identical to the substring with the same length starting from the position h_i. Lines with $h_i = 0$ does not tell any hints except that y_i in the line indicates the end of the substring specified in the line immediately above.

Each of the following q lines has an integer z_i ($1 \leq z_i \leq n$), specifying the position of the letter in the secret string to output.

It is ensured that there exists at least one secret string that matches all the given information. In other words, the given hints have no contradiction.

Output

The output should be a single line consisting only of q characters. The character at position i of the output should be the letter at position z_i of the the secret string if it is uniquely determined from the hints, or `?` otherwise.

Sample Input 1

```
9 4 5 4
3 C
4 I
7 C
9 P
2 1
4 0
6 2
7 0
8 4
8
1
9
6
```

Sample Output 1

```
ICPC
```

Sample Input 2

```
1000000000 1 1 2
20171217 A
3 1
42
987654321
```

Sample Output 2

```
?A
```

Problem K
Counting Cycles
Time Limit: 4 seconds

Given an undirected graph, count the number of simple cycles in the graph. Here, a simple cycle is a connected subgraph all of whose vertices have degree exactly two.

Input

The input consists of a single test case of the following format.

$$
\begin{aligned}
&n\ m\\
&u_1\ v_1\\
&\vdots\\
&u_m\ v_m
\end{aligned}
$$

A test case represents an undirected graph G.

The first line shows the number of vertices n ($3 \leq n \leq 100\,000$) and the number of edges m ($n - 1 \leq m \leq n + 15$). The vertices of the graph are numbered from 1 to n.

The edges of the graph are specified in the following m lines. Two integers u_i and v_i in the i-th line of these m lines mean that there is an edge between vertices u_i and v_i. Here, you can assume that $u_i < v_i$ and thus there are no self loops.

For all pairs of i and j ($i \neq j$), either $u_i \neq u_j$ or $v_i \neq v_j$ holds. In other words, there are no parallel edges.

You can assume that G is connected.

Output

The output should be a line containing a single number that is the number of simple cycles in the graph.

Sample Input 1

```
4 5
1 2
1 3
1 4
2 3
3 4
```

Sample Output 1

```
3
```

Sample Input 2

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

Sample Output 2

```
3
```

Sample Input 3

```
4 6
1 2
1 3
1 4
2 3
2 4
3 4
```

Sample Output 3

```
7
```